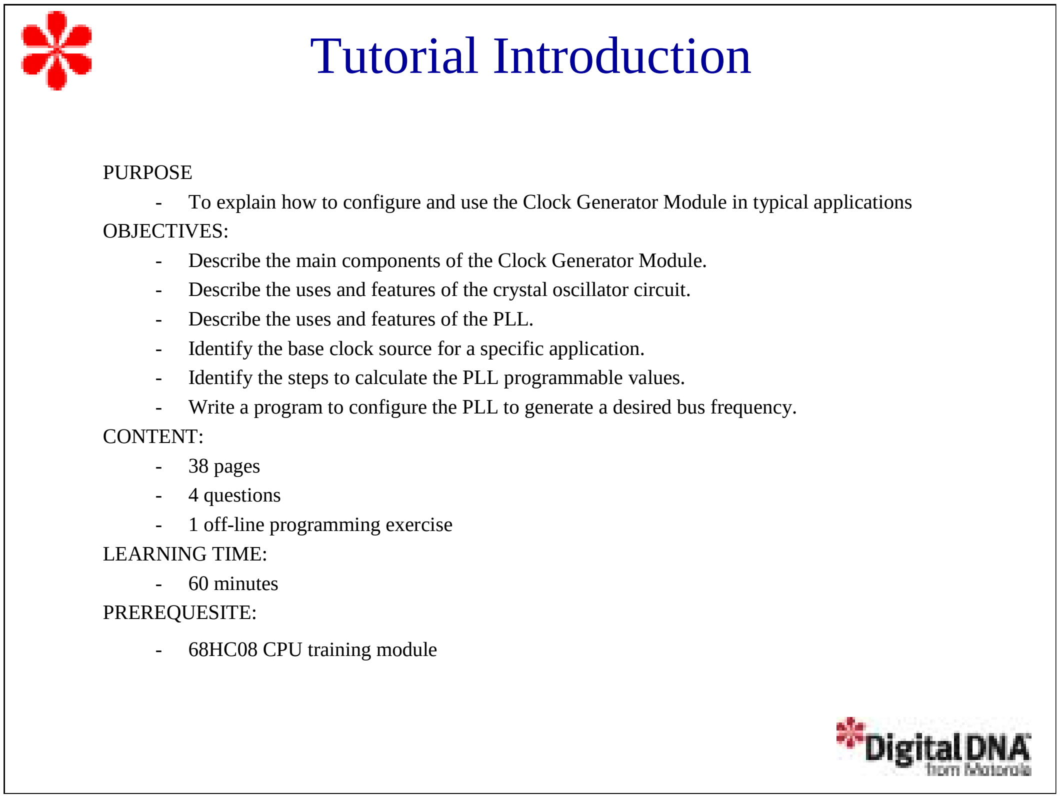


Tutorial Introduction

PURPOSE

- To explain how to configure and use the Clock Generator Module in typical applications

OBJECTIVES:

- Describe the main components of the Clock Generator Module.
- Describe the uses and features of the crystal oscillator circuit.
- Describe the uses and features of the PLL.
- Identify the base clock source for a specific application.
- Identify the steps to calculate the PLL programmable values.
- Write a program to configure the PLL to generate a desired bus frequency.

CONTENT:

- 38 pages
- 4 questions
- 1 off-line programming exercise

LEARNING TIME:

- 60 minutes

PREREQUESITE:

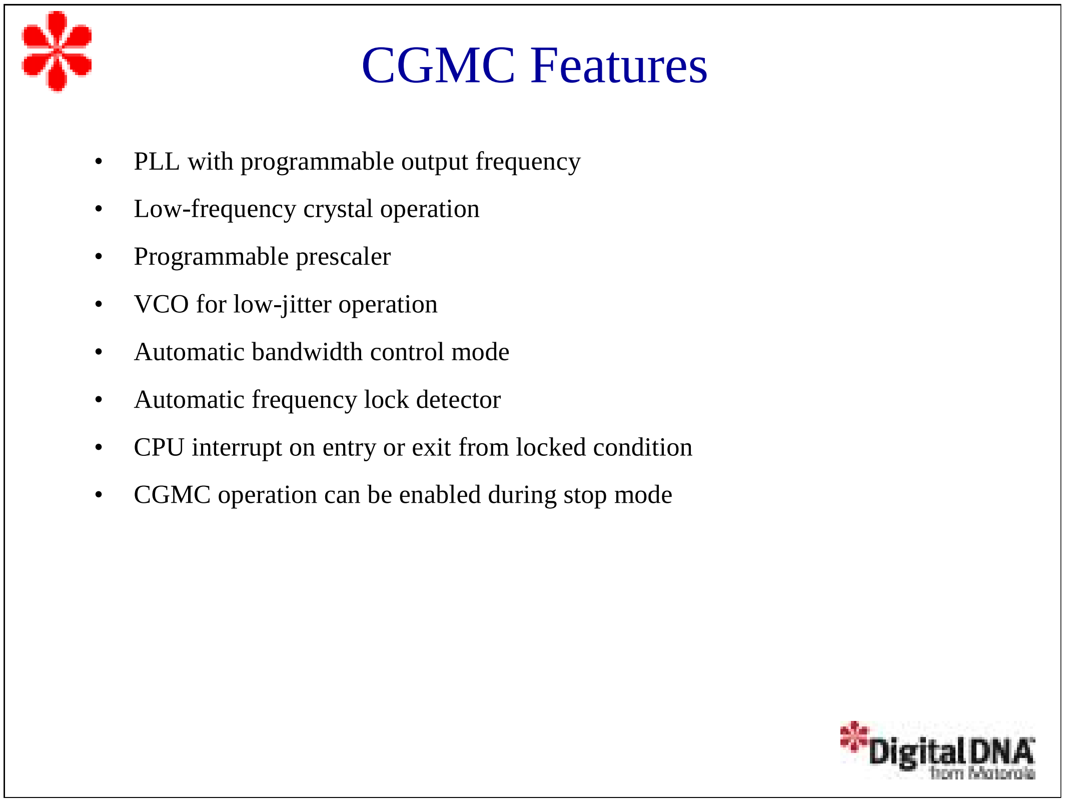
- 68HC08 CPU training module

DigitalDNA from Motorola

Welcome to this tutorial on the 68HC08 Clock Generator Module (CGMC). This tutorial describes the features and configuration of the CGMC. Please note that on subsequent pages, you will find reference buttons in the upper right of the content frame that access additional content.

Upon completion of this tutorial, you'll be able to describe the main components of the CGMC. You'll also be able to select an appropriate base clock source, calculate the different values required to program the phase-locked loop (PLL), and configure the PLL to generate a specific bus frequency.

The recommended prerequisite for this tutorial is the 68HC08 CPU training module. Click the Forward arrow when you're ready to begin the tutorial.



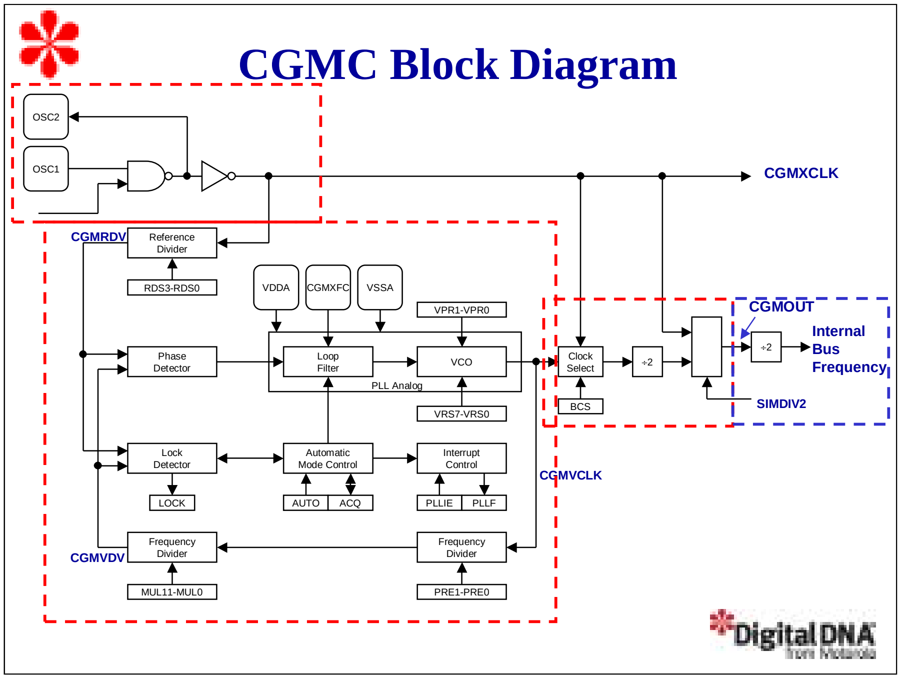
CGMC Features

- PLL with programmable output frequency
- Low-frequency crystal operation
- Programmable prescaler
- VCO for low-jitter operation
- Automatic bandwidth control mode
- Automatic frequency lock detector
- CPU interrupt on entry or exit from locked condition
- CGMC operation can be enabled during stop mode

Let's begin this tutorial with a discussion of the CGMC features. The CGMC includes an integrated PLL that is capable of generating frequencies at integer multiples of the crystal reference. For example, using the PLL can provide up to an 8.2Mhz bus frequency while using a much lower cost 32 Khz crystal. An additional benefit of the PLL is reduced noise generation since high speed crystals and oscillators are not necessary.

The module has two programmable prescalers that provide power-of-two increases in frequency. Also included is a programmable hardware voltage-controlled oscillator, VCO, for low-jitter operation. The CGMC Module can automatically switch from acquisition mode to tracking mode, depending on how far the output is from the reference.

The PLL automatically detects when the output frequency is close enough to the desired frequency and locks. The CPU can be configured to generate an interrupt when the desired frequency is locked or exits from the locked condition. In the CONFIG register, you can set the OSCSTOPENB bit to enable the oscillator to operate during stop mode.



CGMC Block Diagram

The CGMC consists of three major sub-modules: the crystal oscillator circuit, the PLL, and the base clock selector circuit.

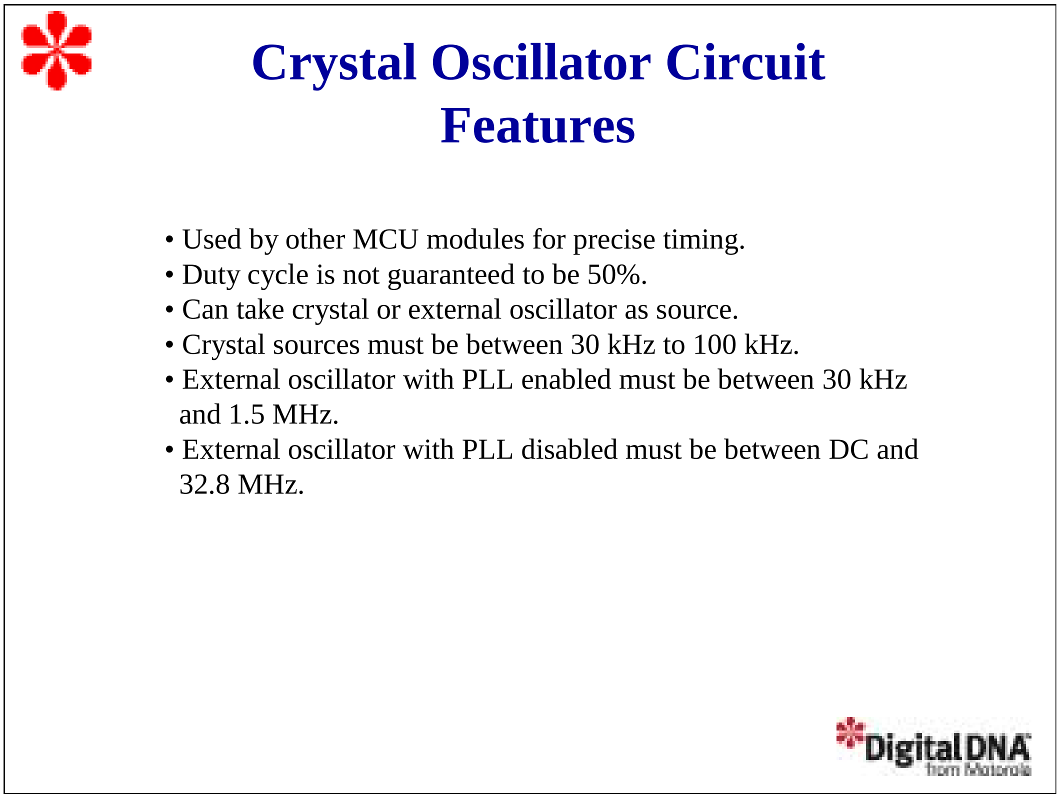
The crystal oscillator circuit consists of an inverting amplifier and an external crystal. The OSC1 pin is the amplifier input and the OSC2 pin is the output. The crystal oscillator circuit generates the constant crystal frequency clock, CGMXCLK, which runs at a rate equal to the crystal frequency. CGMXCLK is also buffered to produce the PLL reference clock.

The PLL is a fully functional frequency generator that is designed for use with crystals or ceramic resonators. The PLL generates the programmable VCO frequency clock, CGMVCLK. The VCO frequency will be an integer multiple of the reference frequency.

The base clock selector circuit is a software-controlled circuit that selects the base clock, CGMOUT. You can set the base clock to either CGMXCLK divided by two or CGMVCLK divided by two.

The System Integration Module (SIM) derives the system clocks from CGMOUT or CGMXCLK with an additional divide by two circuits to produce the internal bus frequency. This will be discussed further in the training.

Let’s take a closer look at each CGMC sub-module, beginning with the crystal oscillator circuit.



Crystal Oscillator Circuit Features

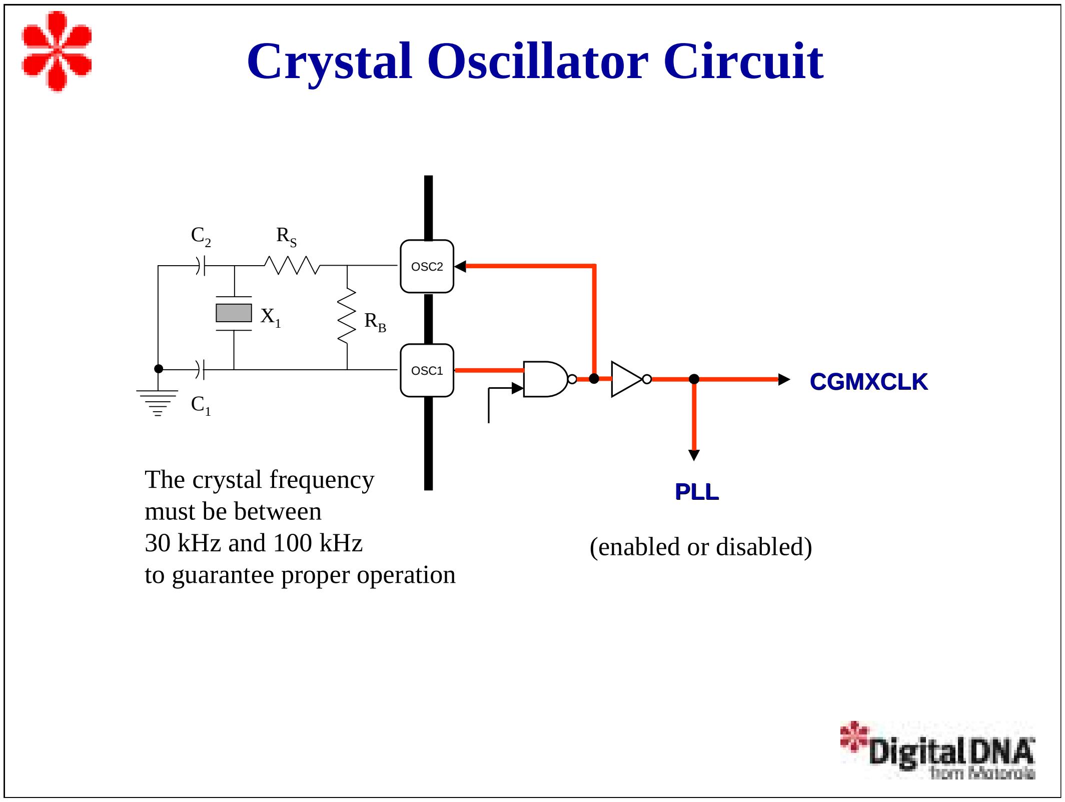
- Used by other MCU modules for precise timing.
- Duty cycle is not guaranteed to be 50%.
- Can take crystal or external oscillator as source.
- Crystal sources must be between 30 kHz to 100 kHz.
- External oscillator with PLL enabled must be between 30 kHz and 1.5 MHz.
- External oscillator with PLL disabled must be between DC and 32.8 MHz.

The CGMXCLK can be used by modules that require precise timing for operation. Note that the duty cycle of CGMXCLK is not guaranteed to be 50% and depends on external factors, including the crystal and related external components. You can also use an externally generated clock to feed the OSC1 pin of the crystal oscillator circuit. To use this configuration, connect the external clock to the OSC1 pin and let the OSC2 pin float.

The crystal oscillator circuit can take two different frequency sources, a crystal or an external oscillator. If you choose to use a crystal, the crystal frequency must be between 30 KHz and 100 KHz, typically 32.768 KHz when using the CGMC PLL. Some 68HC08 derivatives such as the 68HC08 AZ, AB, and MR product Families use the CGMA PLL allowing crystals between 1 MHz and 8 MHz to be used. Check the device databook for differences between the two types of CGM modules.

If you choose to use an external oscillator as the clock source, you can choose to have the PLL enabled or disabled. With the PLL enabled, the external oscillator frequency must be between 30 kHz and 1.5 MHz. With the PLL disabled, the external oscillator frequency can go from direct current, or DC, up to 32 MHz.

Let’s look at the circuit for each of these configurations.

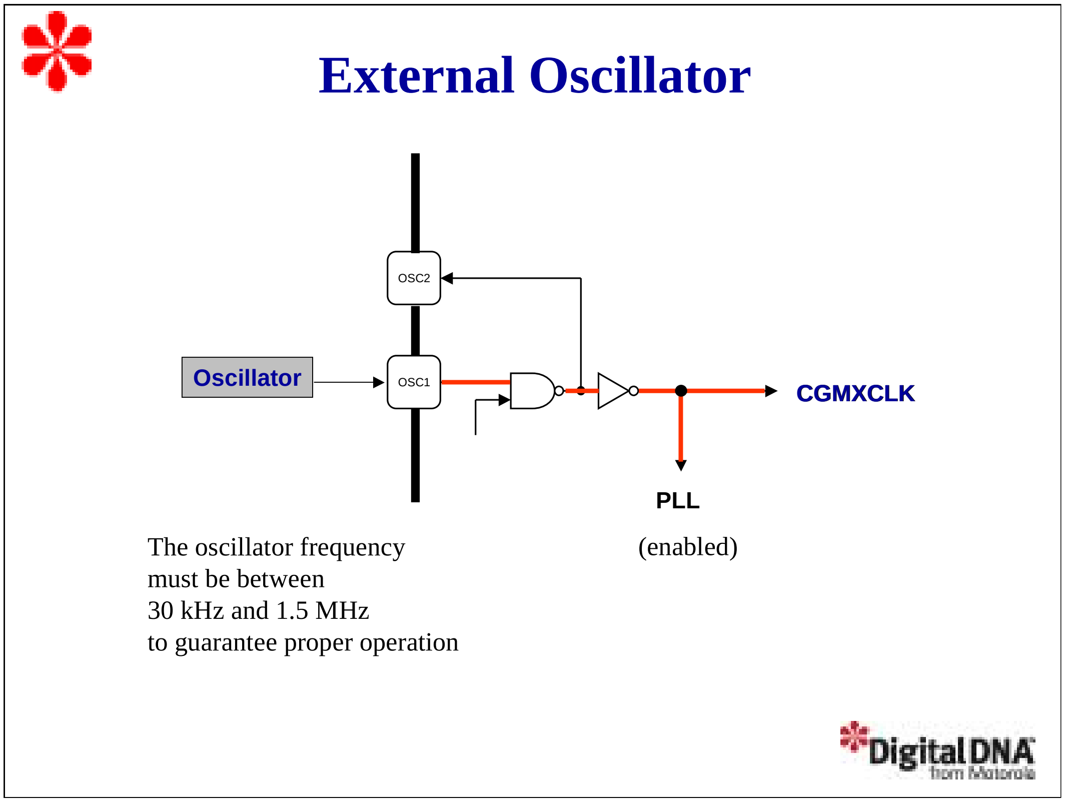


Crystal Oscillator Circuit

The crystal frequency
must be between
30 kHz and 100 kHz
to guarantee proper operation

(enabled or disabled)

When using a crystal as the clock source, the frequency can be selected as the base clock or as a reference clock for the PLL. In this configuration, the PLL may be enabled or disabled.



External Oscillator

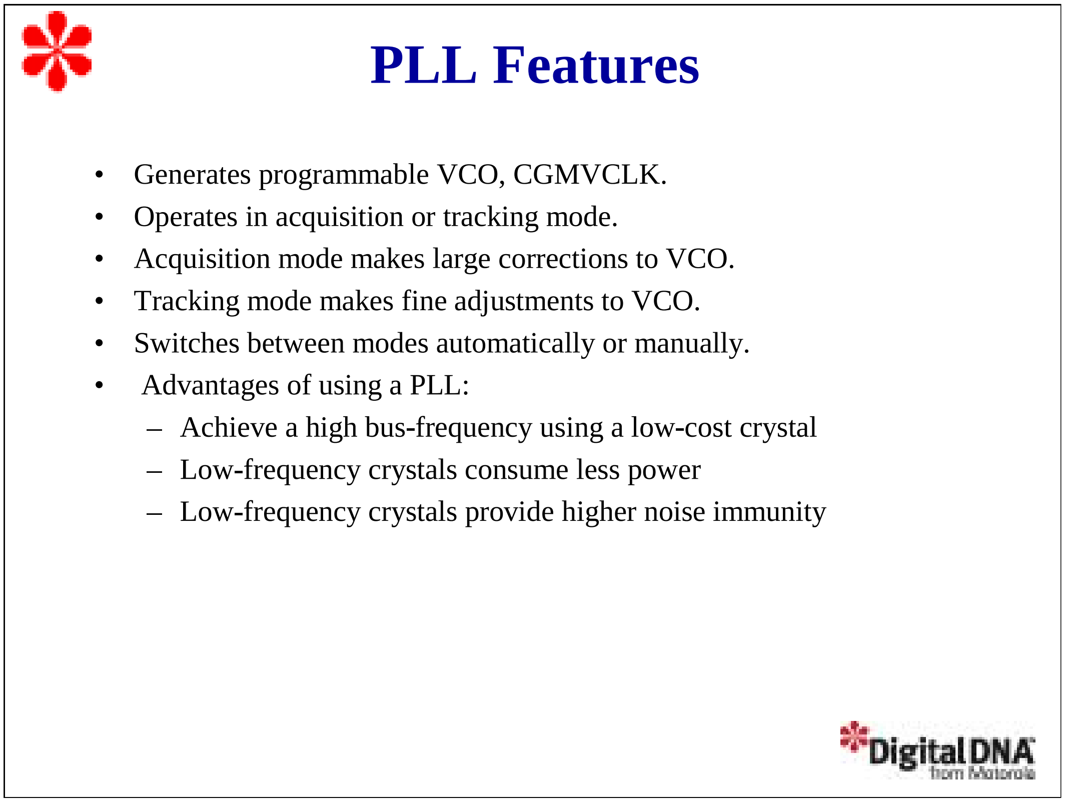
The oscillator frequency
must be between
30 kHz and 1.5 MHz
to guarantee proper operation

(enabled)

With the external oscillator configuration, you can operate with the PLL enabled or disabled depending on the oscillator frequency.

With the PLL enabled, you can use an external oscillator with a frequency between 30Khz and 1.5 Mhz.

With the PLL disabled, you can use an external oscillator with a frequency between DC and 32.8 MHz.



PLL Features

- Generates programmable VCO, CGMVCLK.
- Operates in acquisition or tracking mode.
- Acquisition mode makes large corrections to VCO.
- Tracking mode makes fine adjustments to VCO.
- Switches between modes automatically or manually.
- Advantages of using a PLL:
  - Achieve a high bus-frequency using a low-cost crystal
  - Low-frequency crystals consume less power
  - Low-frequency crystals provide higher noise immunity

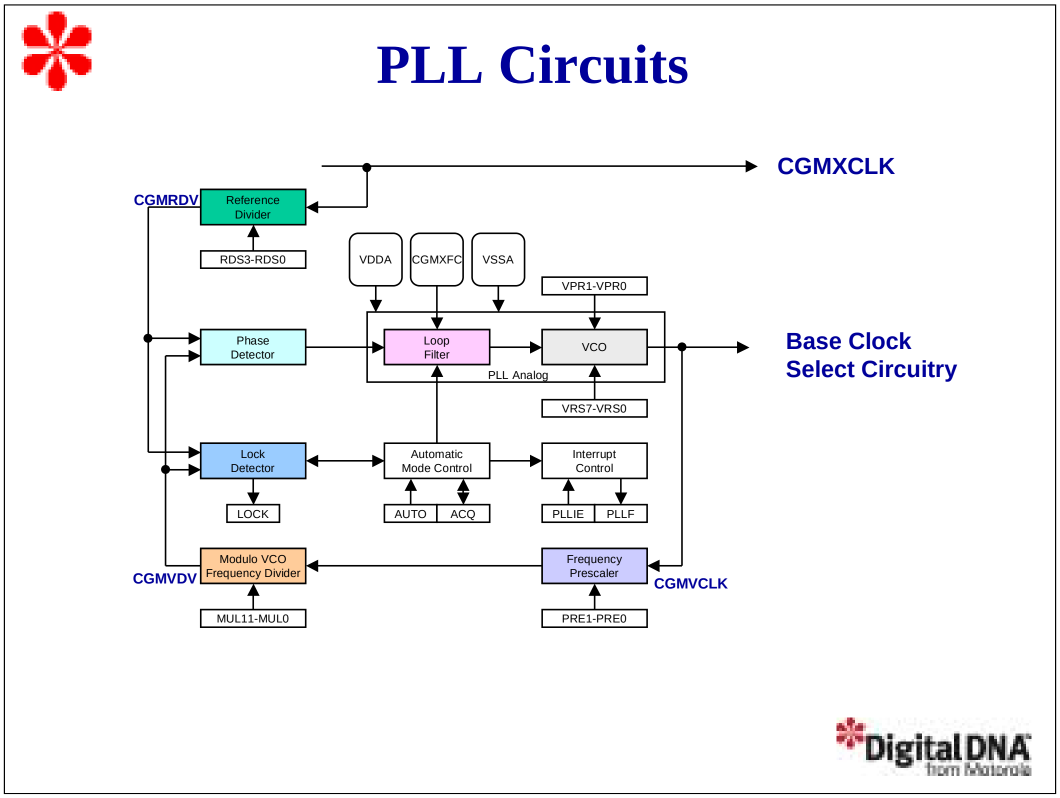
Next, let's take a look at the PLL sub-module. As discussed earlier, the PLL generates the programmable VCO frequency. The PLL can filter the VCO frequency using two modes: acquisition mode and tracking mode. The mode selection depends on the accuracy of the output frequency.

In acquisition mode, the filter can make large frequency corrections to the VCO frequency. This mode is used at PLL startup or when the PLL has suffered a severe noise hit and the VCO frequency is far off the desired frequency. To select this mode, select the ACQ bit in the PLL bandwidth control register.

In tracking mode, the filter makes small corrections to the VCO frequency. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source.

The PLL can change between acquisition and tracking modes either automatically or manually. Automatic mode is recommended for most applications.

There are many advantages to including a PLL in the MCU. You can achieve a high bus frequency using a low-cost crystal, since low-frequency crystals are cheaper. Low-frequency crystals also consume less power and provide higher noise immunity.



PLL Circuits

The PLL is basically an oscillator whose frequency is locked onto a frequency component of an input signal. In this example, the PLL acts as a frequency multiplier to achieve high-output frequency resolution using a low frequency crystal that minimizes cost, power consumption, and noise generation. Let's take a detailed look at how the PLL works.

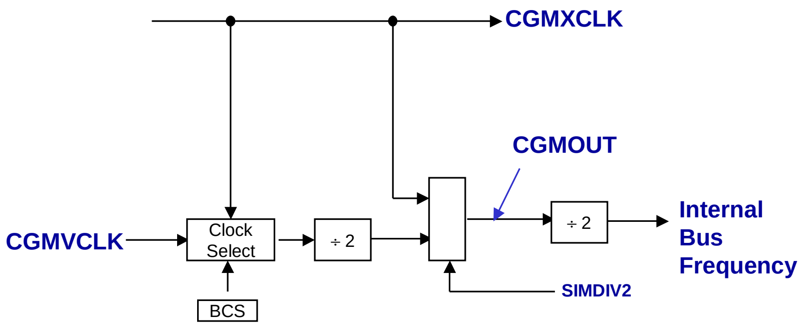
CGMRCLK is the PLL reference clock, a buffered version of CGMXCLK. CGMRCLK runs at a frequency f_{RCLK} and is fed to the PLL through a programmable modulo reference divider, which divides the frequency by a factor R. The divider's output is the final reference clock, CGMRDV.

The VCO output clock, running at a frequency, f_{VCLK} , is fed back through a programmable prescaler divider and a programmable modulo divider. The prescaler divides the VCO clock by a power-of-two factor, P, and the modulo divider reduces this frequency by a factor N. The output is the VCO feedback clock, CGMVDV, which runs at a frequency f_{VDV}.

The phase detector then compares CGMVDV with CGMRDV and generates a correction pulse based on the phase difference of the two signals. The loop filter then slightly alters the DC voltage on the external capacitor connected to CGM/XFC based on the width and direction of the correction pulse.

The lock detector also compares CGMVDV and CGMRDV. The frequency is locked when the phase difference between the two signals is very close to zero.

Base Clock Selector Circuit
SIM Clock Divider



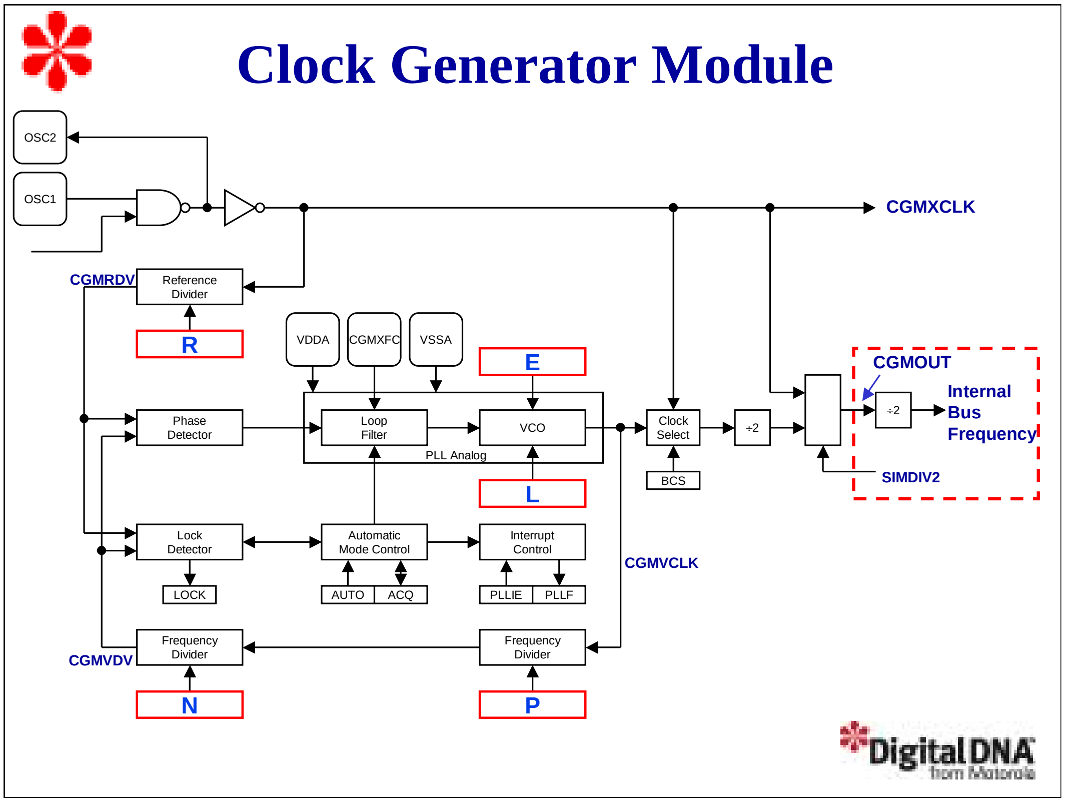
SIMDIV2

Next, let’s look at the last CGMC sub-module, the base clock selector circuit. This circuit is used to select either CGMXCLK or CGMVCLK as the source of the base clock, CGMOUT.

The two input clocks go through a transition control circuit that waits up to three CGMXCLK cycles and three CGMVCLK cycles to change from one clock source to the other. During this time, CGMOUT is held in stasis. The BCS bit in the PLL control register selects which clock drives CGMOUT.

The output of the transition control circuit is then divided by two to correct the duty cycle and to the SIM for an additional divide by two to generate the internal bus frequency, which is one-fourth the frequency of the BCS selected clock (CGMXCLK or CGMVCLK).

A special monitor mode of operation, described in the FLASH training Module, selects CGMXCLK directly through the SIMDIV2 signal to provide an internal bus frequency of CGMXCLK divided by two. Note that this option does not allow the PLL to be used. In normal user mode, the CGMXCLK can only be selected using BCS which results in CGMXCLK divided by 4.



Clock Generator Module

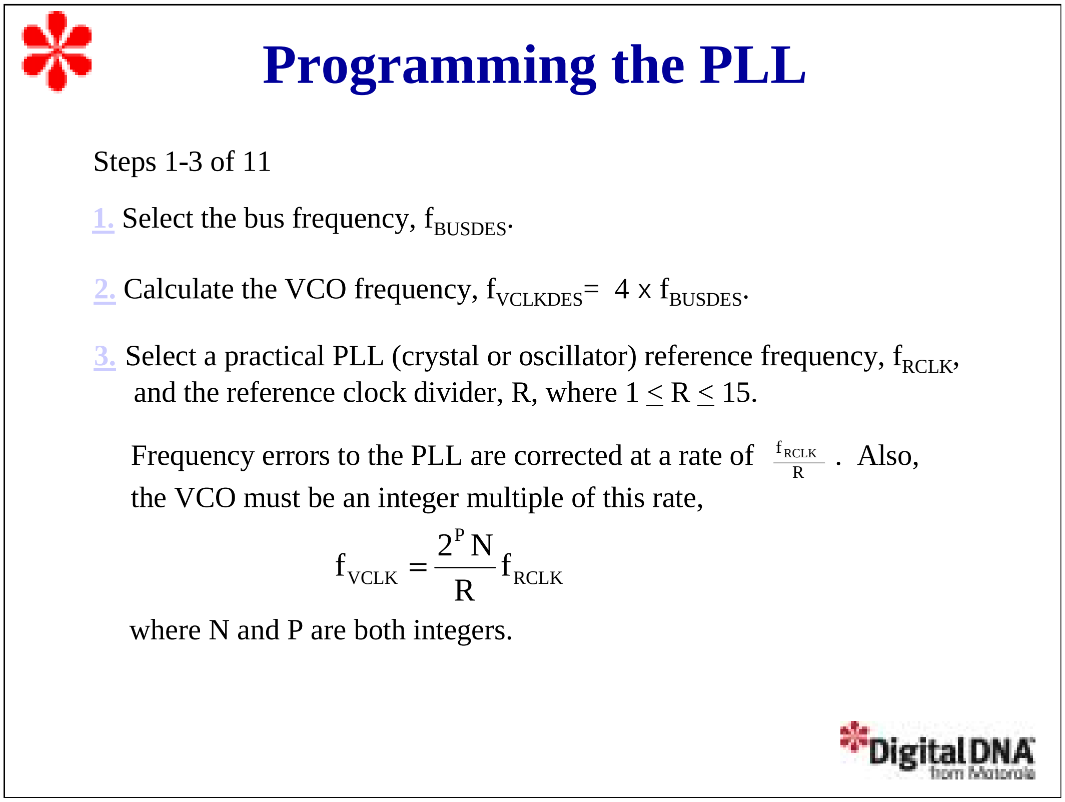
The CGMC uses several factors to program the PLL.

R is the value for the reference divider.

E is the value of the VCO power-of-two range multiplier. L is the VCO linear range multiplier.

P is the prescaler for the frequency divider and N is the value for the modulo VCO frequency divider.

Let’s take a detailed look at how to program the PLL for a specific bus frequency.



Programming the PLL

Steps 1-3 of 11

1. Select the bus frequency, f_{BUSDES}.

2. Calculate the VCO frequency, $f_{VCLKDES} = 4 \times f_{BUSDES}$.

3. Select a practical PLL (crystal or oscillator) reference frequency, f_{RCLK}, and the reference clock divider, R, where $1 \le R \le 15$.

Frequency errors to the PLL are corrected at a rate of $\frac{f_{RCLK}}{R}$. Also, the VCO must be an integer multiple of this rate,

$$f_{VCLK} = \frac{2^P N}{R} f_{RCLK}$$

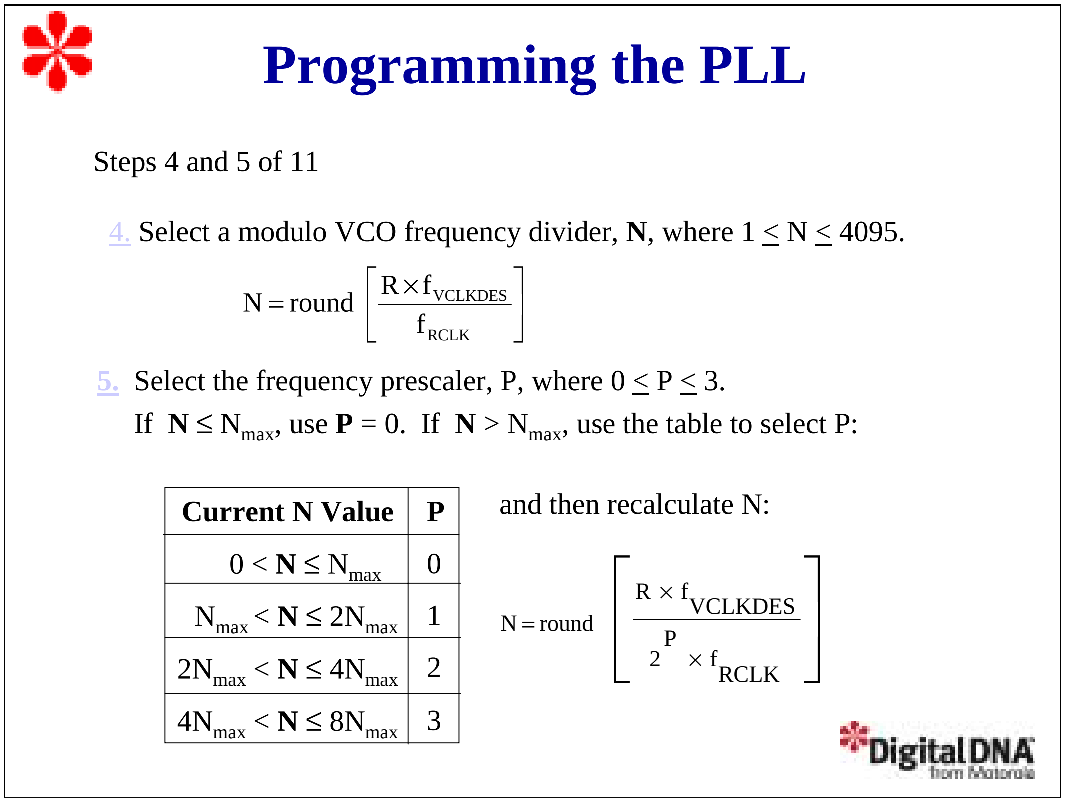
where N and P are both integers.

The first step in programming the PLL is to choose the desired bus frequency, f_{BUSDES}. This is the internal frequency at which the MCU operates. Then calculate the desired VCO frequency, $f_{VCLKDES}$, which is four times the bus frequency.

Next, choose a practical PLL (crystal or oscillator) reference frequency, f_{RCLK}, and the reference clock divider, R.

Note, with an external crystal of 30 kHz - 100 kHz, always set R to 1. With an external high-frequency clock source, between 100 kHz - 1.5 MHz, use R to divide the external frequency to a value between 30 kHz and 100 kHz. Try to choose R as low as possible, since the frequency errors to the PLL are corrected at a rate of f_{RCLK} divided by R. For stability and lock time reduction, this rate must be as fast as possible.

The VCO frequency must be an integer multiple of the rate f_{RCLK} divided by R, as shown.



Programming the PLL

Steps 4 and 5 of 11

4. Select a modulo VCO frequency divider, **N**, where $1 \leq N \leq 4095$.

$$N = \text{round}\left[\frac{R \times f_{VCLKDES}}{f_{RCLK}}\right]$$

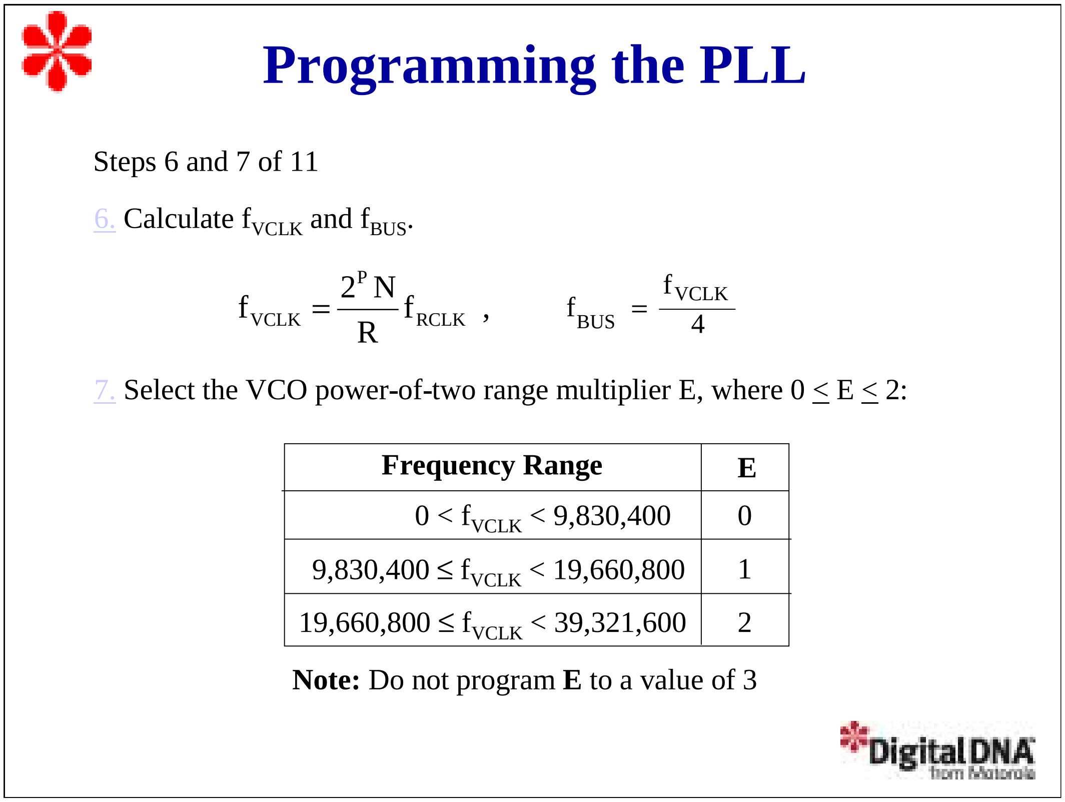
5. Select the frequency prescaler, P, where $0 \leq P \leq 3$.
   If $\mathbf{N} \leq N_{max}$, use **P** = 0. If $\mathbf{N} > N_{max}$, use the table to select P:

Current N Value	P
$0 < \mathbf{N} \leq N_{max}$	0
$N_{max} < \mathbf{N} \leq 2N_{max}$	1
$2N_{max} < \mathbf{N} \leq 4N_{max}$	2
$4N_{max} < \mathbf{N} \leq 8N_{max}$	3

and then recalculate N:

$$N = \text{round}\left[\frac{R \times f_{VCLKDES}}{2^{P} \times f_{RCLK}}\right]$$

Select the modulo frequency divider, N, as follows: multiply R by the VCO frequency, divide the product by the reference frequency, and round the result to the nearest integer. Then select the frequency prescaler, P, based on the value of N. If N is less than or equal to 4095, use a P value of 0. If N is greater than 4095, use the table to select P and recalculate N with the new formula as shown.



Programming the PLL

Steps 6 and 7 of 11

6. Calculate f_{VCLK} and f_{BUS}.

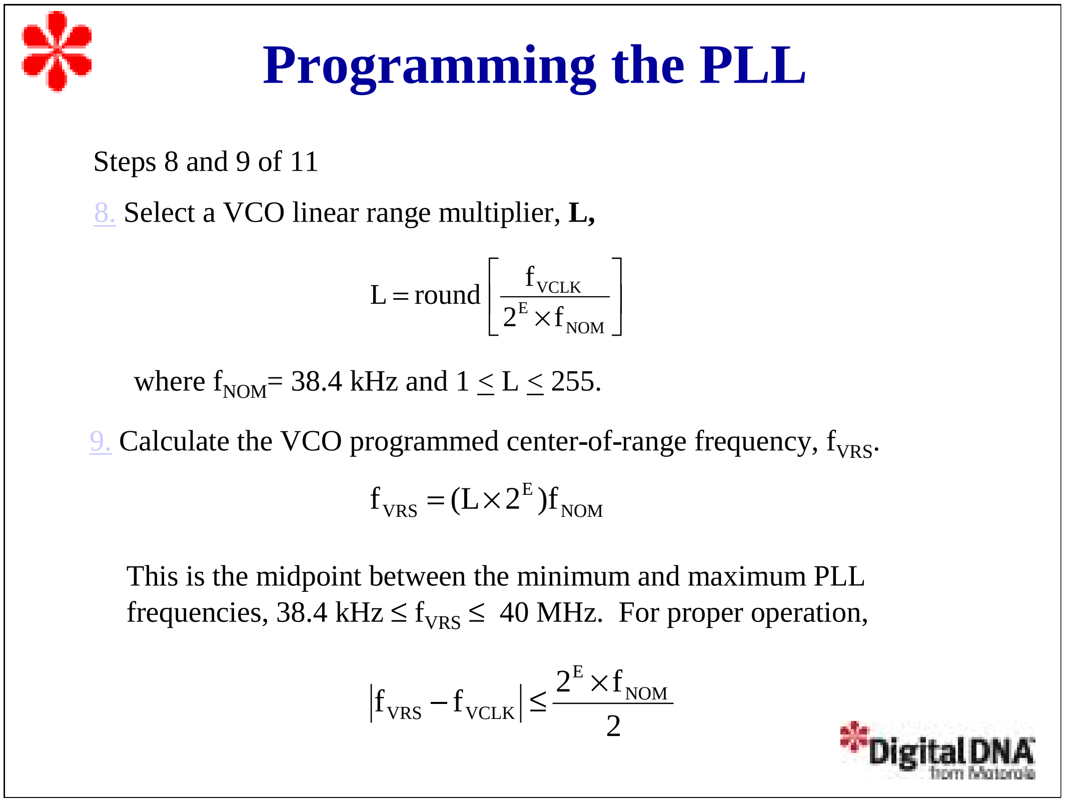
$$f_{VCLK} = \frac{2^P N}{R} f_{RCLK} \quad , \qquad f_{BUS} = \frac{f_{VCLK}}{4}$$

7. Select the VCO power-of-two range multiplier E, where $0 \leq E \leq 2$:

Frequency Range	E
$0 < f_{VCLK} < 9{,}830{,}400$	0
$9{,}830{,}400 \leq f_{VCLK} < 19{,}660{,}800$	1
$19{,}660{,}800 \leq f_{VCLK} < 39{,}321{,}600$	2

Note: Do not program **E** to a value of 3

Next, calculate and verify the VCO frequency and the bus frequency are in range. Then select the VCO power-of-two range multiplier, E, using the table shown.



Programming the PLL

Steps 8 and 9 of 11

8. Select a VCO linear range multiplier, **L,**

$$L = \text{round}\left[\frac{f_{VCLK}}{2^E \times f_{NOM}}\right]$$

where f_{NOM}= 38.4 kHz and $1 \leq L \leq 255$.

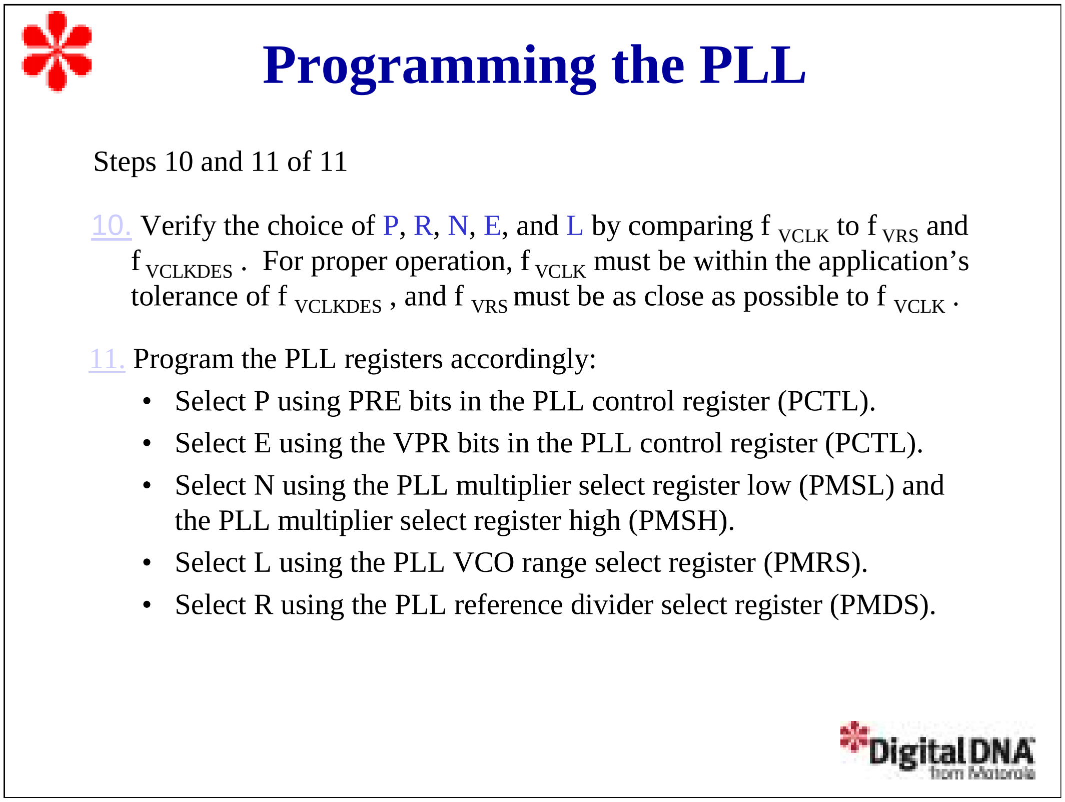
9. Calculate the VCO programmed center-of-range frequency, f_{VRS}.

$$f_{VRS} = (L \times 2^E)f_{NOM}$$

This is the midpoint between the minimum and maximum PLL frequencies, 38.4 kHz ≤ f_{VRS} ≤ 40 MHz. For proper operation,

$$|f_{VRS} - f_{VCLK}| \leq \frac{2^E \times f_{NOM}}{2}$$

Select the VCO linear range multiplier, L, where the nominal center-of-range frequency, f_{NOM}, was selected by design. Then calculate the VCO programmed center-of-range frequency, f_{VRS}. This is the midpoint between the minimum and maximum frequencies attainable by the PLL.



Programming the PLL

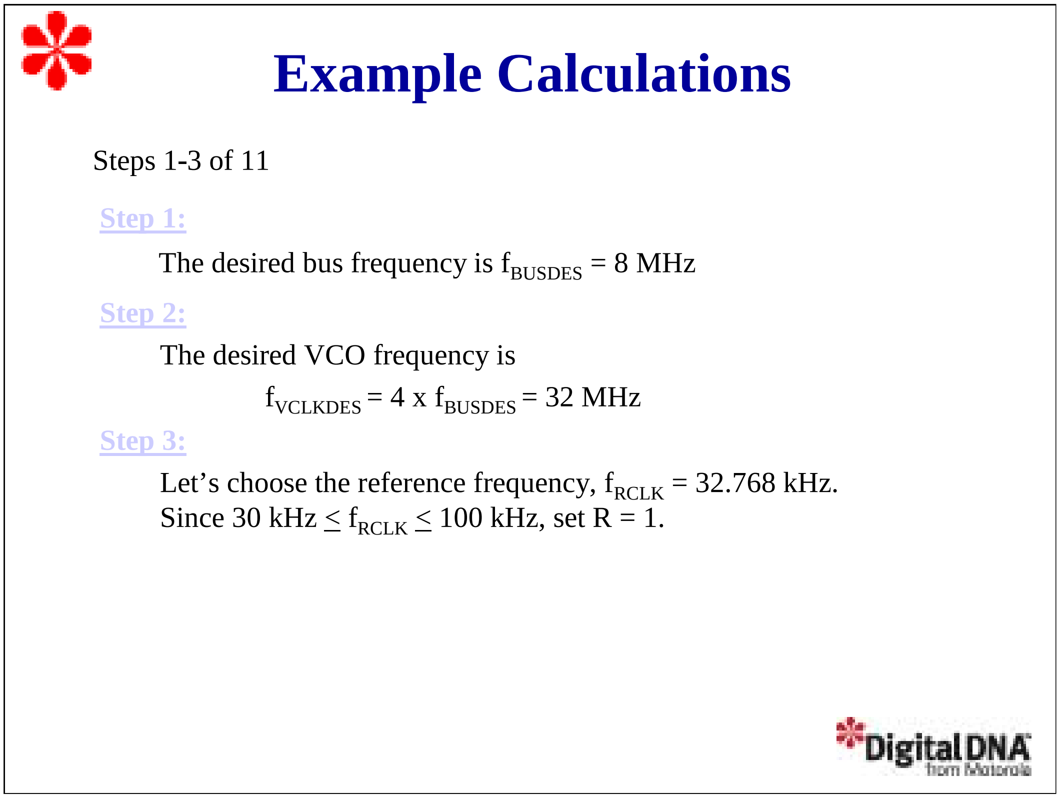
Steps 10 and 11 of 11

10. Verify the choice of P, R, N, E, and L by comparing f_{VCLK} to f_{VRS} and $f_{VCLKDES}$. For proper operation, f_{VCLK} must be within the application's tolerance of $f_{VCLKDES}$, and f_{VRS} must be as close as possible to f_{VCLK}.

11. Program the PLL registers accordingly:
    - Select P using PRE bits in the PLL control register (PCTL).
    - Select E using the VPR bits in the PLL control register (PCTL).
    - Select N using the PLL multiplier select register low (PMSL) and the PLL multiplier select register high (PMSH).
    - Select L using the PLL VCO range select register (PMRS).
    - Select R using the PLL reference divider select register (PMDS).

Digital DNA from Motorola

Verify the choice of P, R, N, E, and L by comparing f_{VCLK} to f_{VRS} and $f_{VCLKDES}$. For proper operation, f_{VCLK} must be within the application's tolerance of $f_{VCLKDES}$, and f_{VRS} must be as close as possible to f_{VCLK}. Then select the values using the PLL registers as shown.



Example Calculations

Steps 1-3 of 11

Step 1:

The desired bus frequency is f_{BUSDES} = 8 MHz

Step 2:

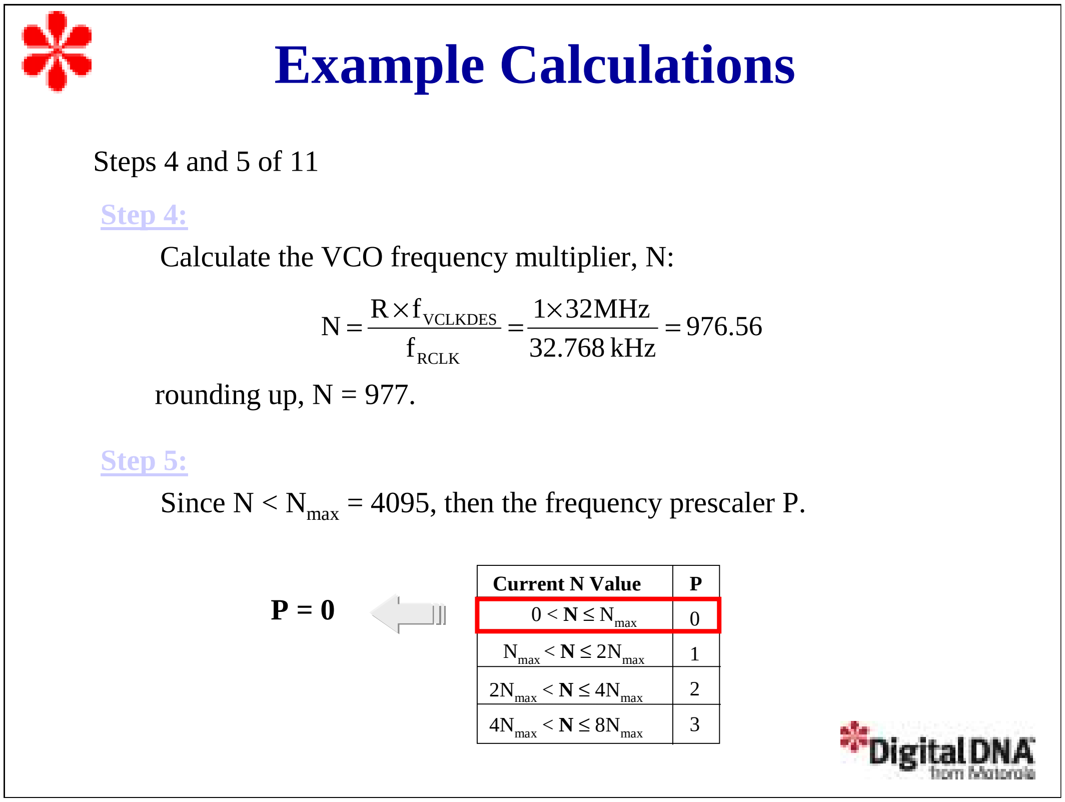
The desired VCO frequency is

$$f_{VCLKDES} = 4 \times f_{BUSDES} = 32 \text{ MHz}$$

Step 3:

Let’s choose the reference frequency, f_{RCLK} = 32.768 kHz.
Since 30 kHz $\leq f_{RCLK} \leq$ 100 kHz, set R = 1.

Now, let’s look at a specific example to demonstrate how simple this process is. Let’s work through the calculations to program the PLL to generate a bus frequency of 8 MHz with a crystal of 32.768 kHz. Since the reference frequency is between 30 kHz and 100 kHz, then the reference divider should be set to 1.



Example Calculations

Steps 4 and 5 of 11

Step 4:

Calculate the VCO frequency multiplier, N:

$$N = \frac{R \times f_{VCLKDES}}{f_{RCLK}} = \frac{1 \times 32\text{MHz}}{32.768\,\text{kHz}} = 976.56$$

rounding up, N = 977.

Step 5:

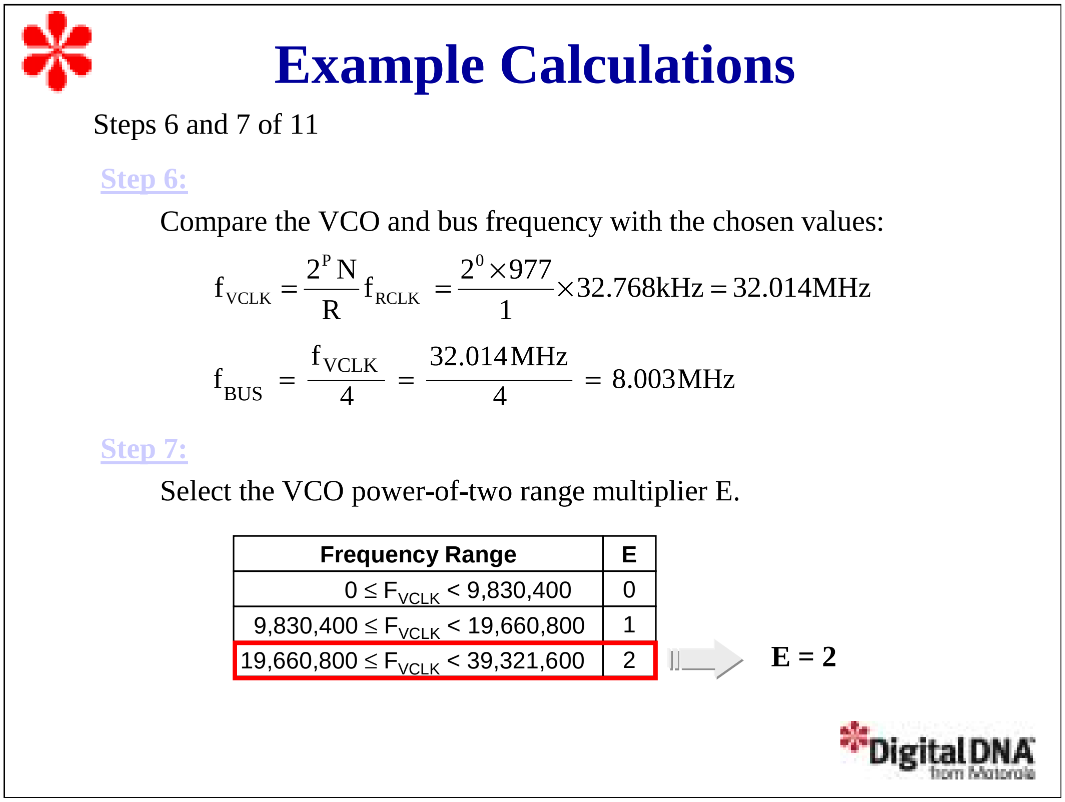
Since $N < N_{max} = 4095$, then the frequency prescaler P.

P = 0

Current N Value	P
$0 < \mathbf{N} \leq N_{max}$	0
$N_{max} < \mathbf{N} \leq 2N_{max}$	1
$2N_{max} < \mathbf{N} \leq 4N_{max}$	2
$4N_{max} < \mathbf{N} \leq 8N_{max}$	3

Digital DNA
from Motorola

Using these values, we calculate a VCO frequency multiplier value of 976.56. Rounding this value yields N = 977. Since the value of N is less than the maximum value of 4095, we set the prescaler value P = 0.



Example Calculations

Steps 6 and 7 of 11

Step 6:

Compare the VCO and bus frequency with the chosen values:

$$f_{VCLK} = \frac{2^P N}{R} f_{RCLK} = \frac{2^0 \times 977}{1} \times 32.768\text{kHz} = 32.014\text{MHz}$$

$$f_{BUS} = \frac{f_{VCLK}}{4} = \frac{32.014\text{MHz}}{4} = 8.003\text{MHz}$$

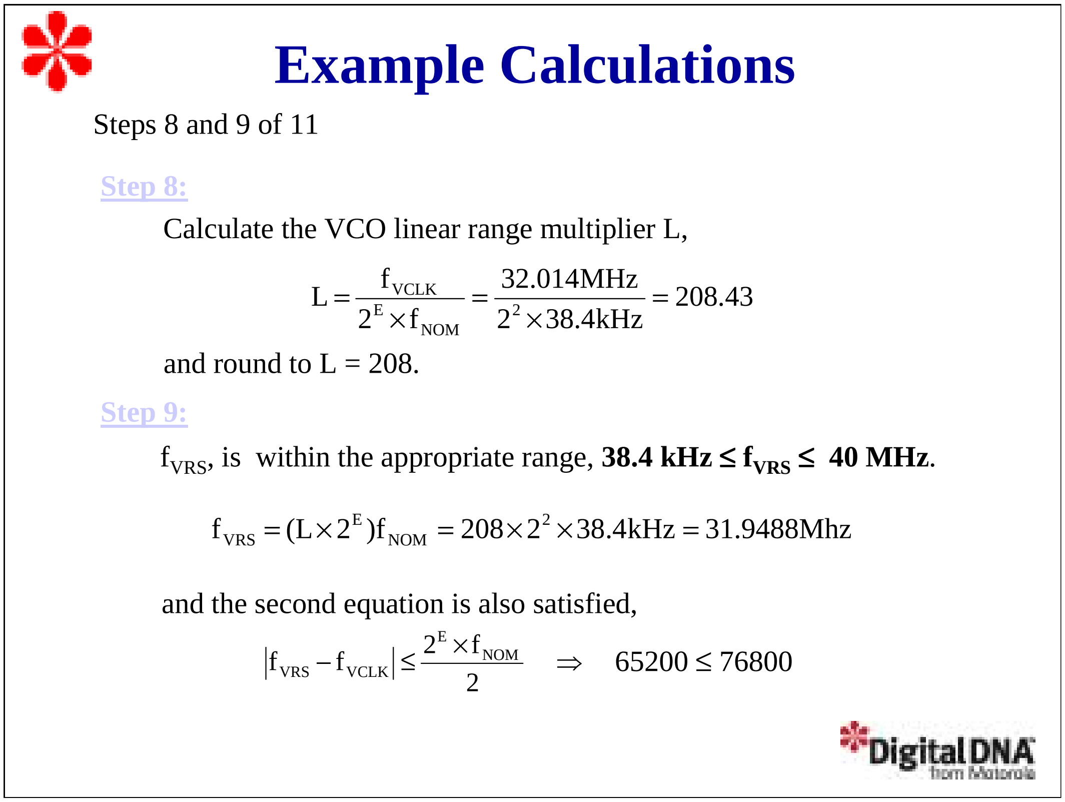
Step 7:

Select the VCO power-of-two range multiplier E.

Frequency Range	E
$0 \le F_{VCLK} < 9{,}830{,}400$	0
$9{,}830{,}400 \le F_{VCLK} < 19{,}660{,}800$	1
$19{,}660{,}800 \le F_{VCLK} < 39{,}321{,}600$	2

E = 2

Next, verify the values of f_{VCLK} and f_{BUS} by comparing them to the desired values of 32 MHz and 8 MHz respectively. Our calculations produced a VCO frequency of 32.014 MHz and bus frequency of 8.003 MHz. These values are very close to what we wanted. The small differences were introduced when we rounded the value of N to get an integer value.



Example Calculations

Steps 8 and 9 of 11

Step 8:

Calculate the VCO linear range multiplier L,

$$L = \frac{f_{VCLK}}{2^E \times f_{NOM}} = \frac{32.014\text{MHz}}{2^2 \times 38.4\text{kHz}} = 208.43$$

and round to L = 208.

Step 9:

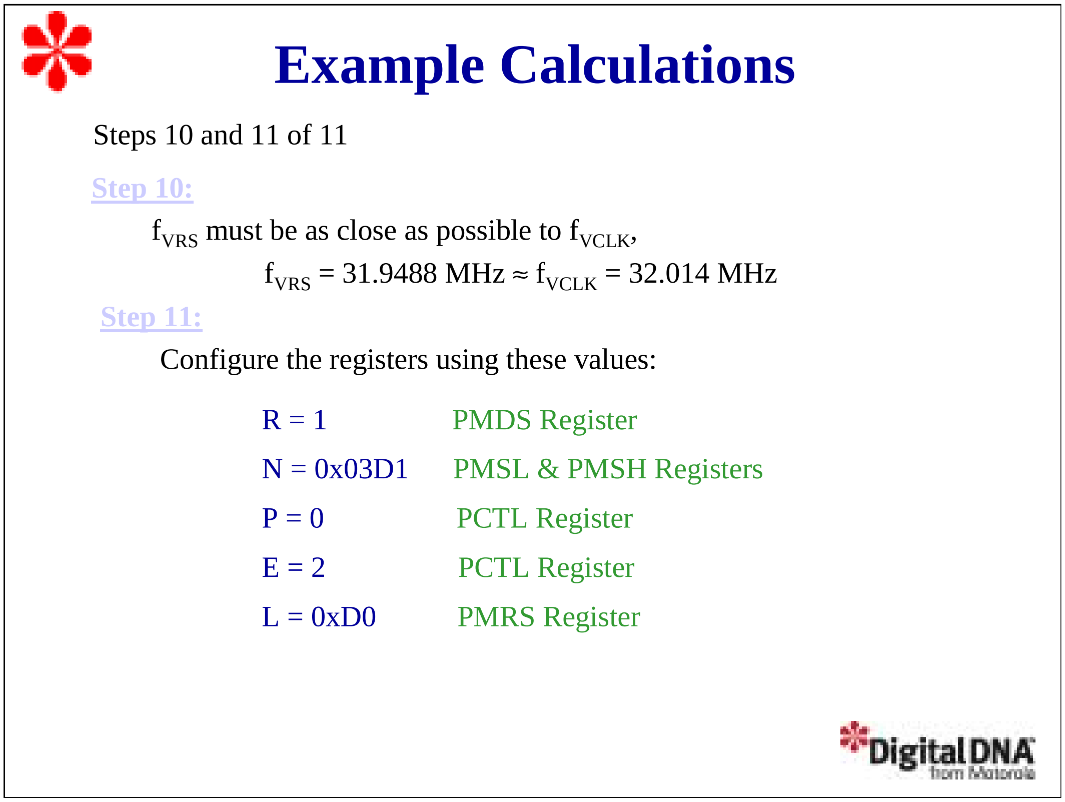
f_{VRS}, is within the appropriate range, **38.4 kHz ≤ f_{VRS} ≤ 40 MHz**.

$$f_{VRS} = (L \times 2^E) f_{NOM} = 208 \times 2^2 \times 38.4\text{kHz} = 31.9488\text{Mhz}$$

and the second equation is also satisfied,

$$\left| f_{VRS} - f_{VCLK} \right| \le \frac{2^E \times f_{NOM}}{2} \quad \Rightarrow \quad 65200 \le 76800$$

Calculate the VCO linear range multiplier and round the result to L = 208. f_{VRS} is within the appropriate range and satisfies the second equation.



Example Calculations

Steps 10 and 11 of 11

Step 10:

f_{VRS} must be as close as possible to f_{VCLK},

$$f_{VRS} = 31.9488 \text{ MHz} \approx f_{VCLK} = 32.014 \text{ MHz}$$

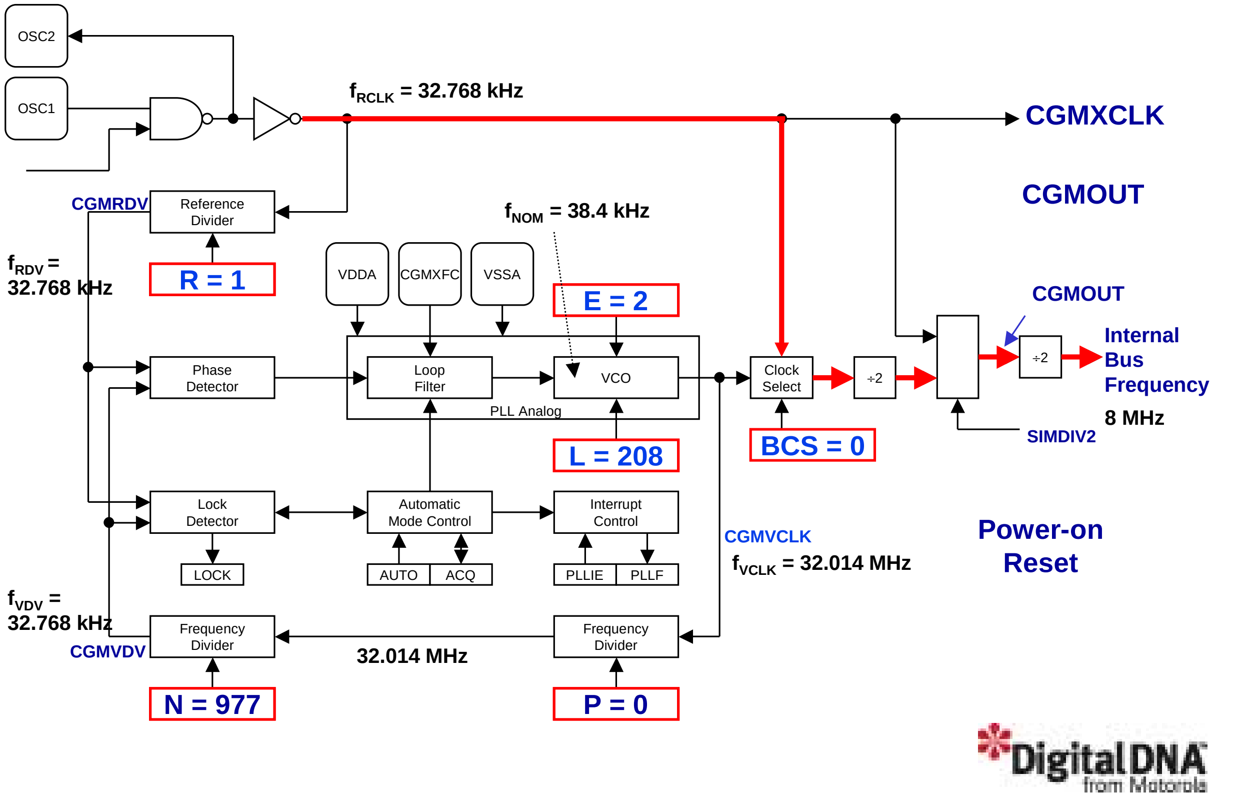
Step 11:

Configure the registers using these values:

R = 1	PMDS Register
N = 0x03D1	PMSL & PMSH Registers
P = 0	PCTL Register
E = 2	PCTL Register
L = 0xD0	PMRS Register

Verify the choice of P, R, N, E, and L by comparing f_{VCLK} to f_{VRS} and $f_{VCLKDES}$. Then configure the PLL registers with the values R = 1, N = 0x03D1, P = 0, E = 2, and L = 0xD0.

Generating Bus Frequency



Let's look at how the CGM generates the 8MHz bus frequency. We've programmed the registers with the values for R, N, P, E, and L that we calculated in the previous example. Recall that our clock frequency, f_{RCLK}, is 32.768 kHz.

This is fed through the reference divider, which divides f_{RCLK} by the factor R=1, so the reference clock frequency f_{RDV}, is 32.768 kHz.

The VCO nominal center-of-range frequency, f_{NOM}, is 38.4 kHz.

The VCO output clock frequency, f_{VCLK} is affected by the linear factor L=208 and the power-of-two factor E=2. After several passes, f_{VCLK} becomes 32.014 MHz.

Then f_{VCLK} goes through a power-of-two prescaler P=0, so f_{VCLK} divided by 2^P is 32.014MHz.

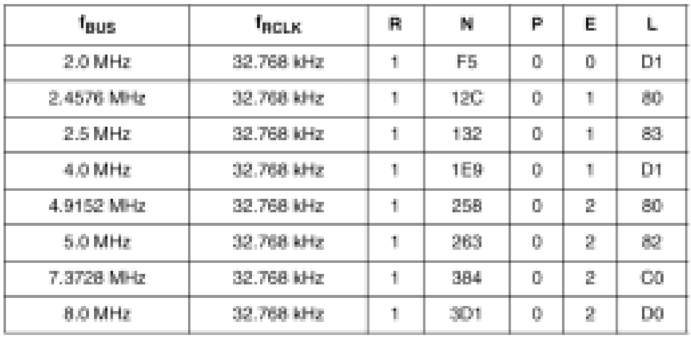
Now the VCO clock is reduced by a factor N, getting f_{VDV} = 32.768 kHz. This frequency is compared with our reference clock, f_{RCLK}. Once it matches the PLL will be locked. With BCS set to one, the Internal bus frequency will be our target 8 Mhz.

Now, let's look at what happens when a power-on reset occurs. After a power on reset, the VCO clock is not locked and the BCS bit is cleared.

This selects the CMGXCLK as the base clock, which is 32.768 kHz.

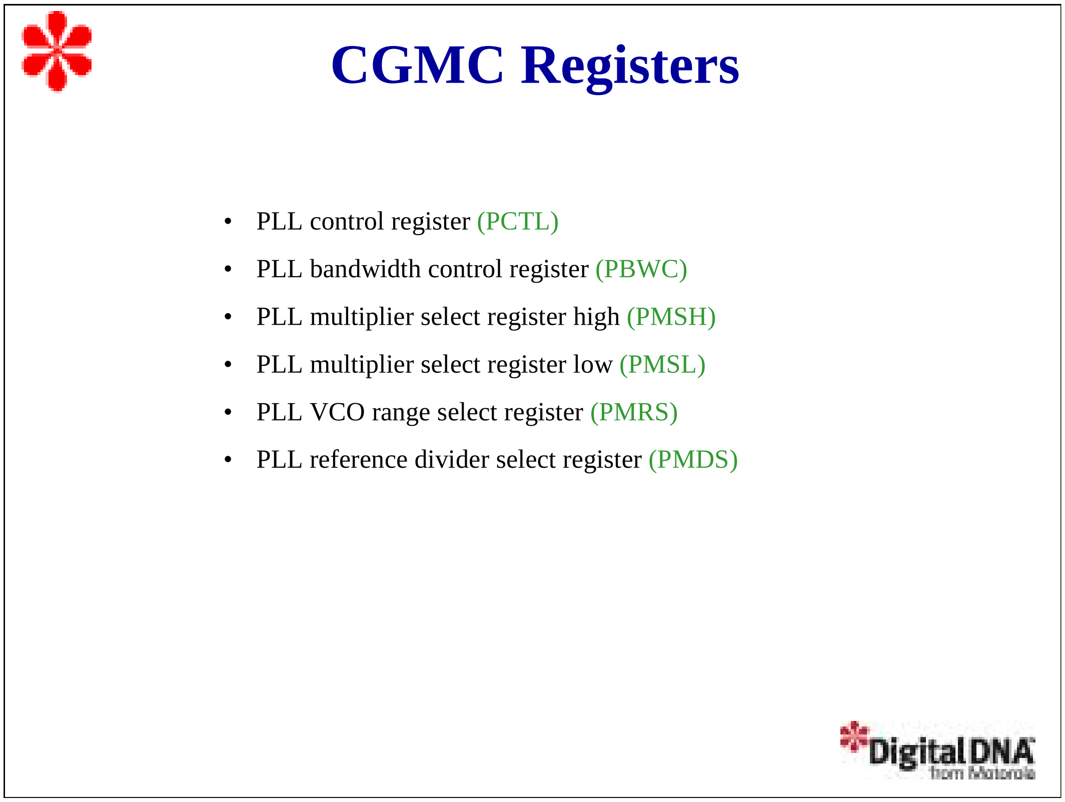
Then, until the VCO clock signal is locked, the bus frequency is 8.192 kHz instead of the 8MHz target frequency.

Precalculated Frequencies



f_{BUS}	f_{RCLK}	R	N	P	E	L
2.0 MHz	32.768 kHz	1	F5	0	0	D1
2.4576 MHz	32.768 kHz	1	12C	0	1	80
2.5 MHz	32.768 kHz	1	132	0	1	83
4.0 MHz	32.768 kHz	1	1E9	0	1	D1
4.9152 MHz	32.768 kHz	1	258	0	2	80
5.0 MHz	32.768 kHz	1	263	0	2	82
7.3728 MHz	32.768 kHz	1	384	0	2	C0
8.0 MHz	32.768 kHz	1	3D1	0	2	D0

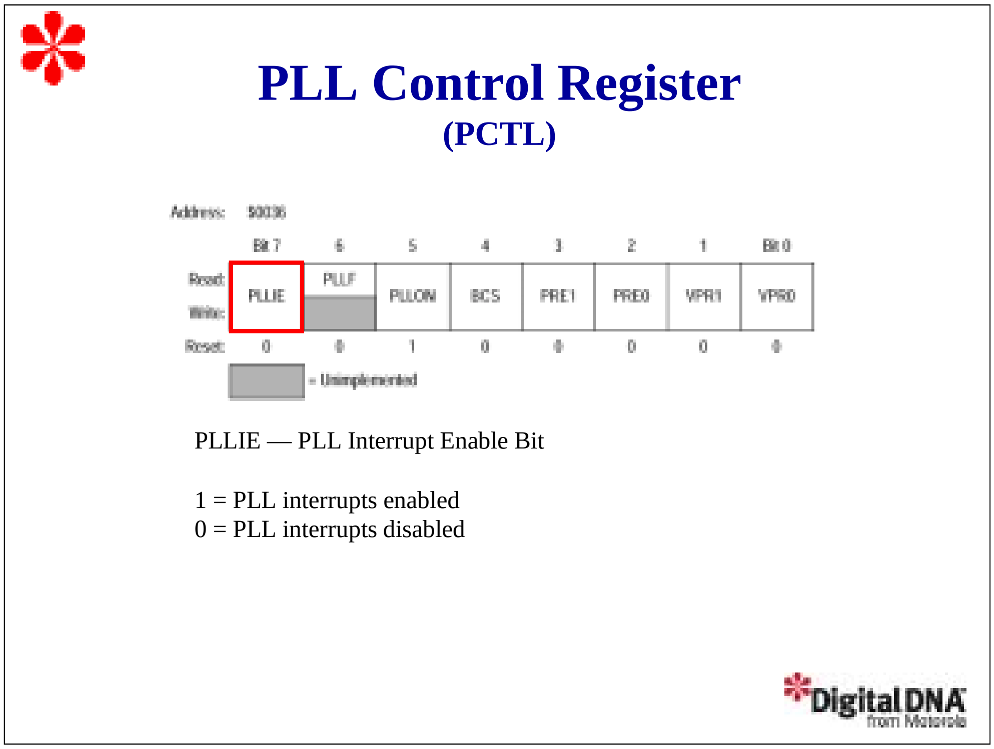
Motorola provides a table with some examples for different bus frequencies using a typical 32.768 kHz crystal. This makes it easy to program the PLL for typical bus frequencies. Note that the values are already in hexadecimal notation.



CGMC Registers

- PLL control register (PCTL)
- PLL bandwidth control register (PBWC)
- PLL multiplier select register high (PMSH)
- PLL multiplier select register low (PMSL)
- PLL VCO range select register (PMRS)
- PLL reference divider select register (PMDS)

Next, let’s discuss the CGMC register set which includes six registers for configuring and monitoring CGMC operations. We’ll take a detailed look at each register, beginning with the PLL control register, PCTL.



PLL Control Register

(PCTL)

Address: $0036

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
Write:		▨						
Reset:	0	0	1	0	0	0	0	0

▨ = Unimplemented

PLLIE — PLL Interrupt Enable Bit

1 = PLL interrupts enabled
0 = PLL interrupts disabled

Let’s examine the PCTL register. As we’ll see, the PCTL is used to program the values of P and E.

The PLL interrupt enable bit, PLL, is a read/write bit that enables the PLL to generate an interrupt request when the LOCK bit toggles, setting the PLL flag, PLLF. When the AUTO bit in the PLL bandwidth control register (PBWC) is clear, PLLIE cannot be written and reads as logic 0. Reset clears the PLLIE bit.

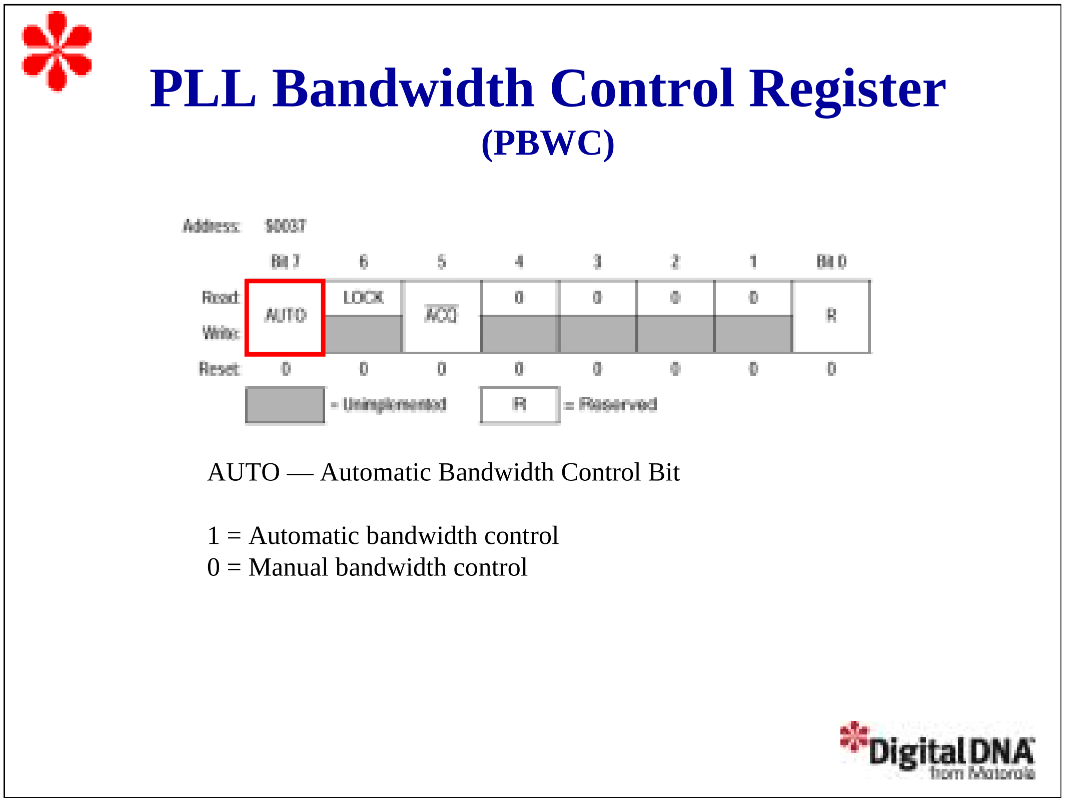
The PLL interrupt flag bit, PLLF, is a read-only bit that is set whenever the LOCK bit toggles. PLLF generates an interrupt request if the PLLIE bit also is set. PLLF always reads as logic 0 when the AUTO bit in the PLL bandwidth control register (PBWC) is clear. The PLLF bit is cleared on reset or when you read the PCTL register.

The PLL on bit, PLLON, is a read/write bit that activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). Reset sets this bit so that the loop can stabilize as the MCU is powering up.

The base clock select bit, BCS, is a read/write bit that selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of CGMOUT. The CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling the BCS bit, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. Reset clears the BCS bit.

The prescaler program bits, PRE1-PRE0, are read/write bits that select the prescaler power-of-two multiplier, P. PRE1 and PRE0 cannot be written to when the PLLON bit is set. Reset clears these bits. Note that the value of P is typically 0 when using a 32.768-kHz crystal as the reference clock.

The power-of-two range select bits, VPR1-VPR0, are read/write bits that control the VCO’s hardware power-of-two range multiplier E that, in conjunction with L, controls the hardware center-of-range frequency, f_{VRS}. Note that these bits cannot be written when the PLLON bit is set. Reset clears these bits.



PLL Bandwidth Control Register

(PBWC)

Address: $0037

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	AUTO	LOCK	ACQ	0	0	0	0	R
Write:		▨		▨	▨	▨	▨	
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

AUTO — Automatic Bandwidth Control Bit

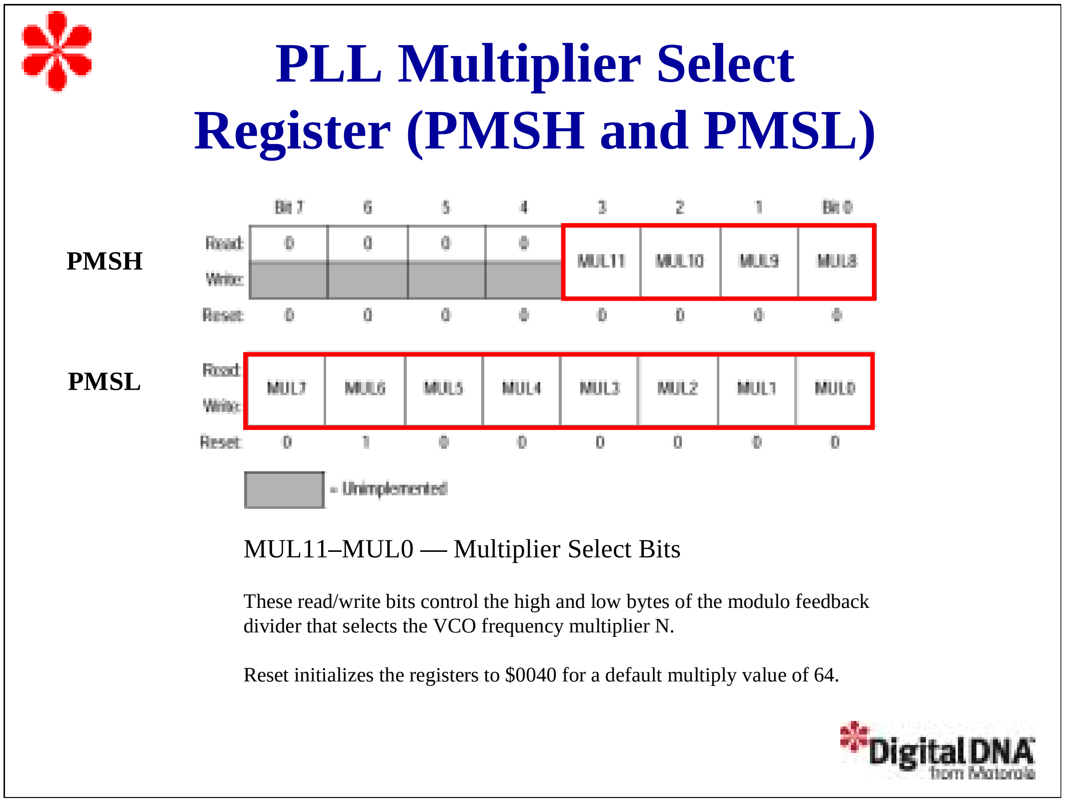
1 = Automatic bandwidth control
0 = Manual bandwidth control

The PLL bandwidth control register, PBWC, is used to configure the PLL mode selection. It is also used to indicate when the PLL is locked.

The automatic bandwidth control bit, AUTO, is a read/write bit that selects automatic or manual bandwidth control. When initializing the PLL for manual operation (AUTO = 0), clear the ACQ bit before turning on the PLL. Reset clears the AUTO bit.

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as logic 0 and has no meaning. The writing to this bit is reserved for testing, so this bit must always be written as 0. Reset clears the LOCK bit.

When the AUTO bit is set, the acquisition mode bit, ACQ, is a read-only bit that indicates whether the PLL is in acquisition mode or tracking mode. When the AUTO bit is clear, ACQ is a read/write bit that controls whether the PLL is in acquisition or tracking mode. In automatic bandwidth control mode (AUTO = 1), the last-written value from manual operation is stored in a temporary location and is recovered when manual operation resumes. Reset clears this bit and enables acquisition mode.



PLL Multiplier Select Register (PMSH and PMSL)

PMSH

		Bit 7	6	5	4	3	2	1	Bit 0
PMSH	Read:	0	0	0	0	MUL11	MUL10	MUL9	MUL8
	Write:	▨	▨	▨	▨				
	Reset:	0	0	0	0	0	0	0	0
PMSL	Read: Write:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
	Reset:	0	1	0	0	0	0	0	0

▨ = Unimplemented

MUL11–MUL0 — Multiplier Select Bits

These read/write bits control the high and low bytes of the modulo feedback divider that selects the VCO frequency multiplier N.

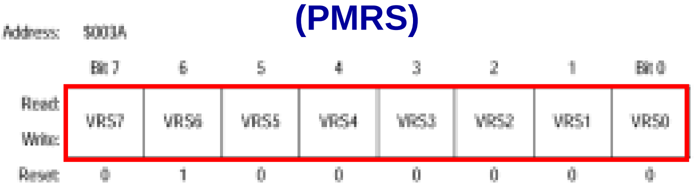
Reset initializes the registers to $0040 for a default multiply value of 64.

The PLL multiplier select register is a two-byte register used to program the value of the VCO frequency multiplier, N. The multiplier select bits, MUL11-MUL0, are read/write bits that control the high and low bytes of the modulo feedback divider that selects the VCO frequency divider. Bits 7 - 4 of the high byte are unimplemented and always read as logic 0. Writing a value of $0000 to these registers configures the modulo feedback divider the same as writing a value of $0001. Reset initializes the registers to $0040 for a default multiply value of 64.

Note that the multiplier select bits have built-in protection such that they cannot be written to when the PLL is on (PLLON = 1).

PLL VCO Range Select Register

(PMRS)



Address: $003A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
Reset:	0	1	0	0	0	0	0	0

VRS7–VRS0 — VCO Range Select Bits

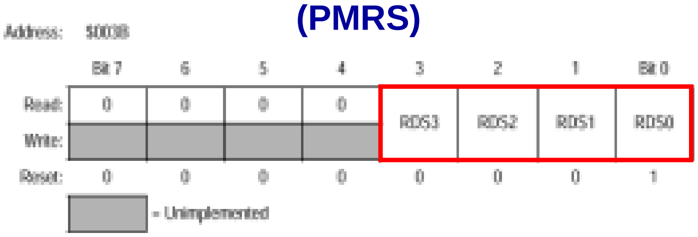
These read/write bits control the hardware center-of-range linear multiplier L. A value of $00 disables the PLL and clears the BCS bit in the PCTL.

Reset initializes the register to $40 for a default range multiply value of 64.

The PLL VCO range select register, PMRS, contains the information required to program the VCO linear range multiplier L that, in conjunction with E, controls the hardware center-of-range frequency, f_{VRS}. The VCO range select bits, VRS7–VRS0, cannot be written to when the PLLON bit in the PCTL register is set. Reset initializes the register to $40 for a default range multiply value of 64.

PLL VCO Range Select Register

(PMRS)



Address: $003B

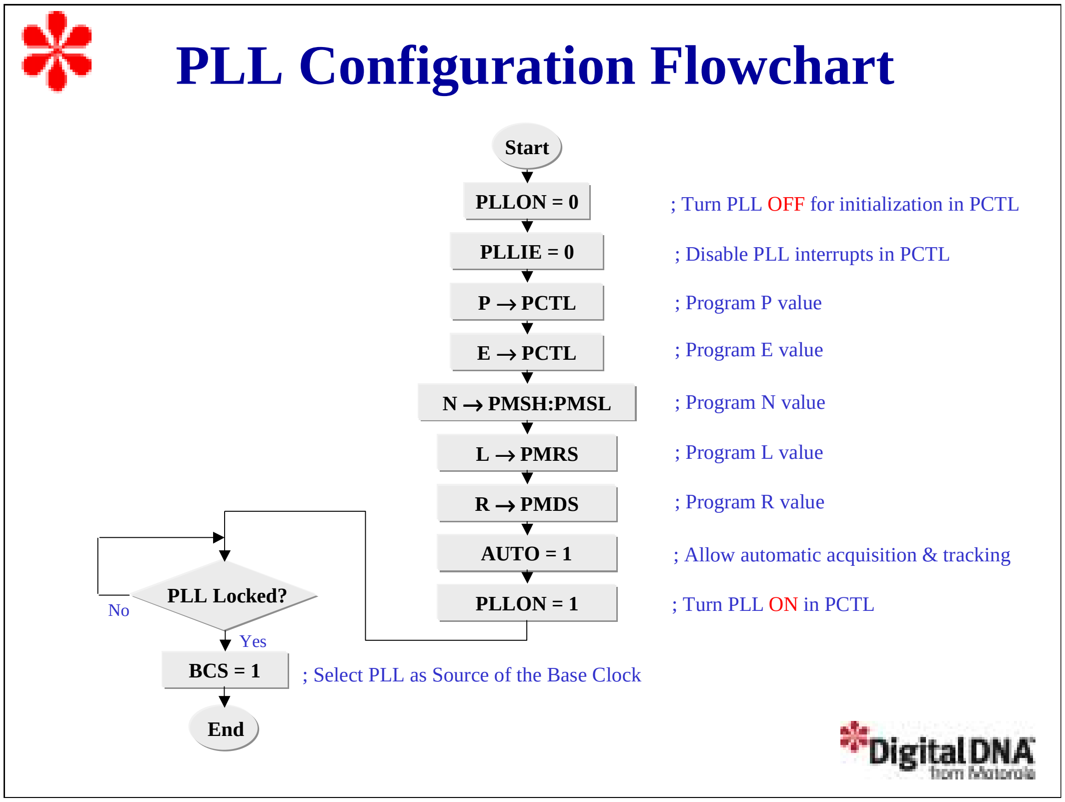
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	RDS3	RDS2	RDS1	RDS0
Write:	▨	▨	▨	▨				
Reset:	0	0	0	0	0	0	0	1

▨ = Unimplemented

RDS3–RDS0 — Reference Divider Select Bits

Selects the modulo reference divider that programs the reference division factor, R. Reset selects default divide value of 1.

The PLL reference divider select Register, PMDS, contains the programming information required to configure the reference clock divider, R. The reference divider select bits, RDS7-RDS0, cannot be written to when the PLL is on. Writing a value of $00 to the PMDS register configures the reference divider the same as a value of $01. Reset initializes the register to $01 for a default divider value of 1.



PLL Configuration Flowchart

Start

PLLON = 0 ; Turn PLL OFF for initialization in PCTL

PLLIE = 0 ; Disable PLL interrupts in PCTL

P → PCTL ; Program P value

E → PCTL ; Program E value

N → PMSH:PMSL ; Program N value

L → PMRS ; Program L value

R → PMDS ; Program R value

AUTO = 1 ; Allow automatic acquisition & tracking

PLLON = 1 ; Turn PLL ON in PCTL

PLL Locked?

No

Yes

BCS = 1 ; Select PLL as Source of the Base Clock

End

Digital DNA from Motorola

Next, let’s discuss how to configure the PLL using polling.

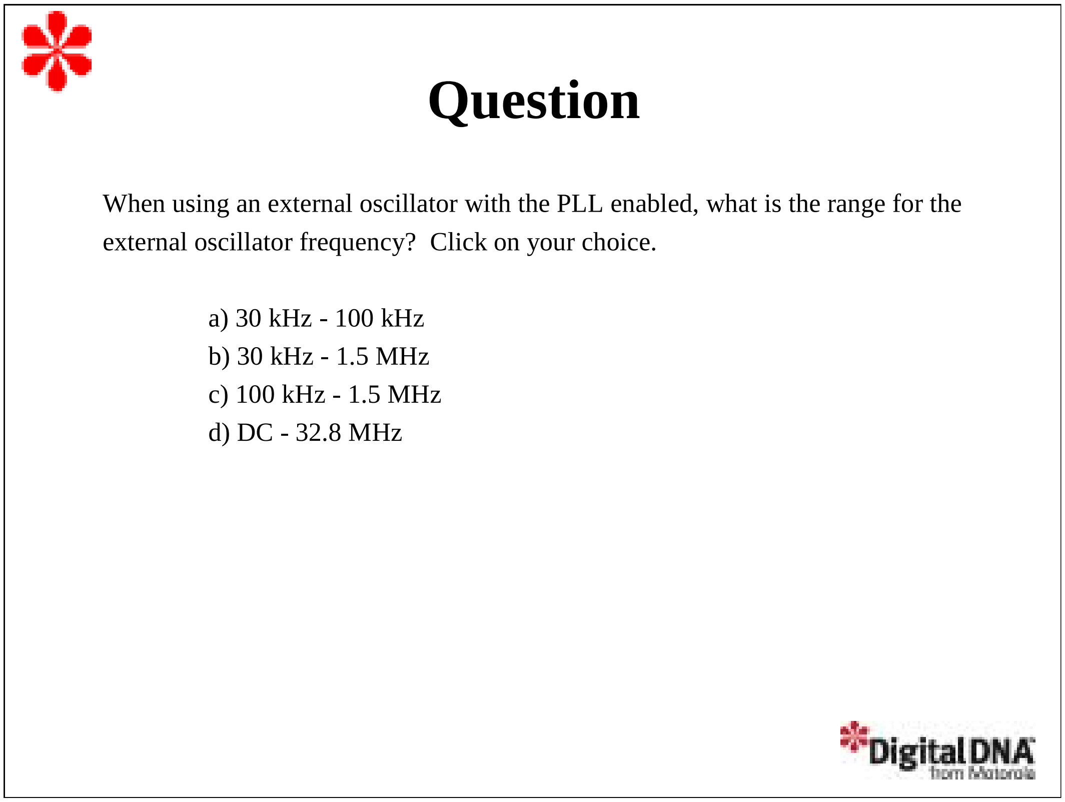
First, turn the PLL OFF and disable the PLL interrupts since we’ll use polling to check the lock.

Then program the P,E,N,L and R values in the CGMC registers.

Set the AUTO bit to allow the loop filter to automatically switch between acquisition and tracking modes.

Turn the PLL ON to reach the desired frequency.

Wait until the VCO frequency has reached the desired value, that is, until the PLL locks the signal. Then select the PLL as the source of the base clock.



Question

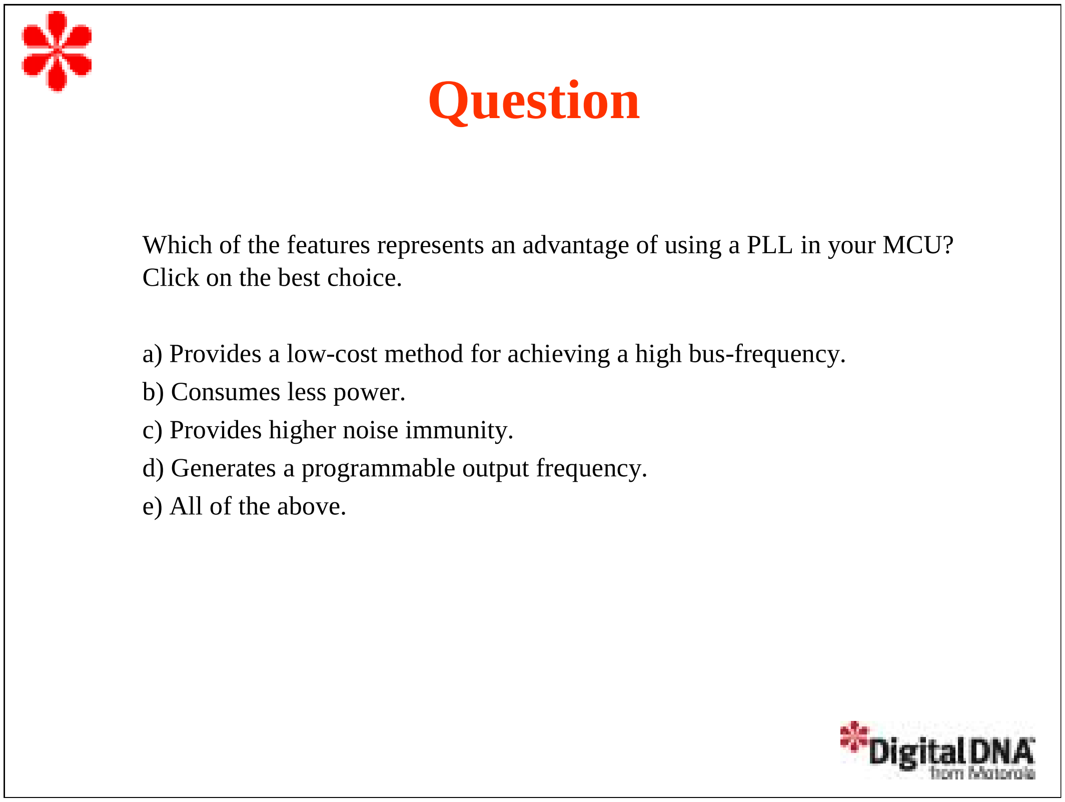
When using an external oscillator with the PLL enabled, what is the range for the external oscillator frequency? Click on your choice.

a) 30 kHz - 100 kHz
b) 30 kHz - 1.5 MHz
c) 100 kHz - 1.5 MHz
d) DC - 32.8 MHz

Let’s consider some questions to check your understanding of the material.

When using an external oscillator with the PLL enabled, what is the range for the external oscillator frequency? Click on your choice.

Answer: The PLL can only work with frequency oscillations between 30 kHz and 1.5 MHz. Therefore, when the PLL is enabled, the frequency range for the external oscillator is choice b.



Question

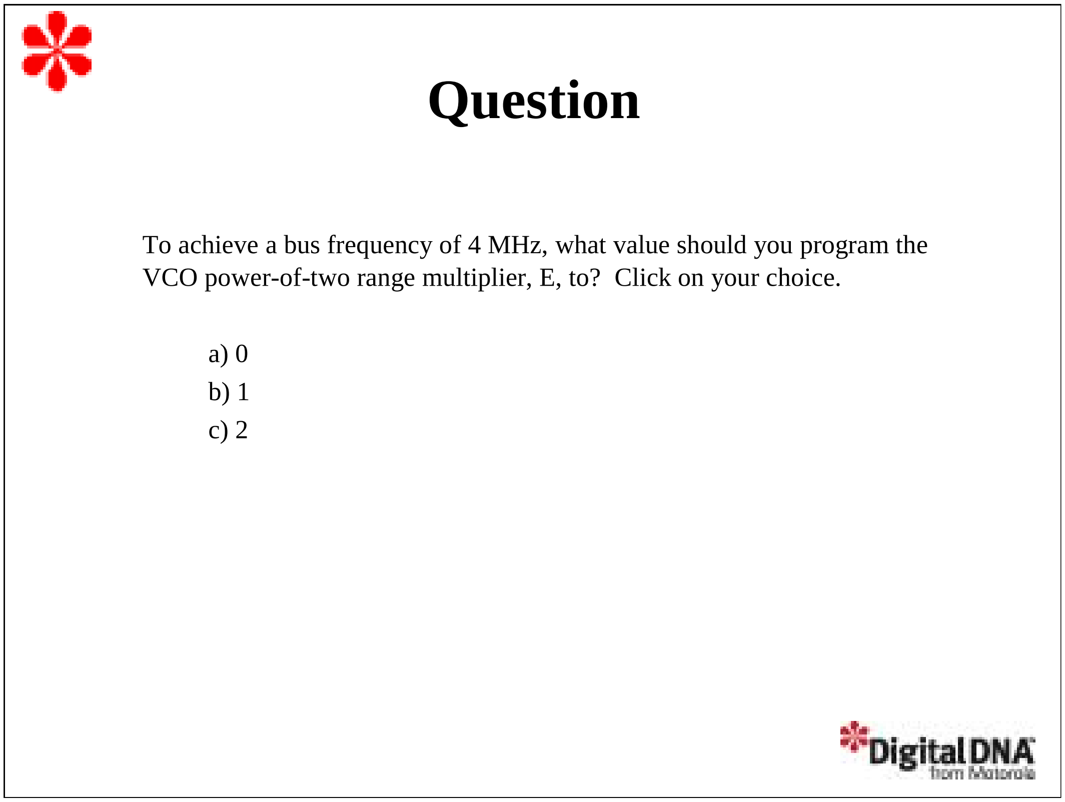
Which of the features represents an advantage of using a PLL in your MCU? Click on the best choice.

a) Provides a low-cost method for achieving a high bus-frequency.
b) Consumes less power.
c) Provides higher noise immunity.
d) Generates a programmable output frequency.
e) All of the above.

DigitalDNA from Motorola

Which of the features represents an advantage of using a PLL in your MCU? Click on your choice.

Answer: All of the features are advantages of using a PLL. The PLL generates a programmable output frequency. Using a low-frequency crystal, the PLL is a low-cost method for achieving a high bus-frequency. Also, low-frequency crystals consume less power and provide higher noise immunity.



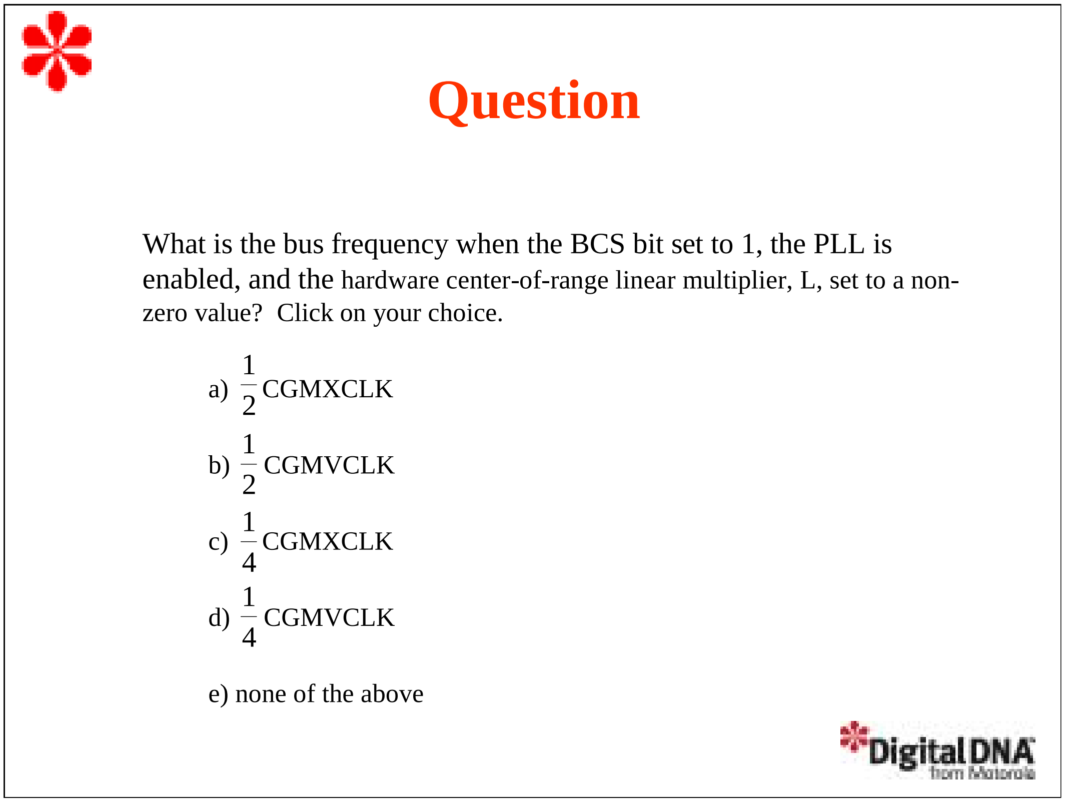
Question

To achieve a bus frequency of 4 MHz, what value should you program the VCO power-of-two range multiplier, E, to? Click on your choice.

a) 0
b) 1
c) 2

To achieve a bus frequency of 4 MHz, what value should you program the VCO power-of-two range multiplier, E, to? Click on your choice.

Answer: We determine the value of E based on the the VCO frequency, f_{VCLK}, which is four times the bus frequency. In this case, the bus frequency is 4 MHz and f_{VCLK} is 16 MHz. Since this value is between the range of 9,830,400 and 19,660,800, the correct value for E is 1.



Question

What is the bus frequency when the BCS bit set to 1, the PLL is enabled, and the hardware center-of-range linear multiplier, L, set to a non-zero value? Click on your choice.

a) $\frac{1}{2}$ CGMXCLK

b) $\frac{1}{2}$ CGMVCLK

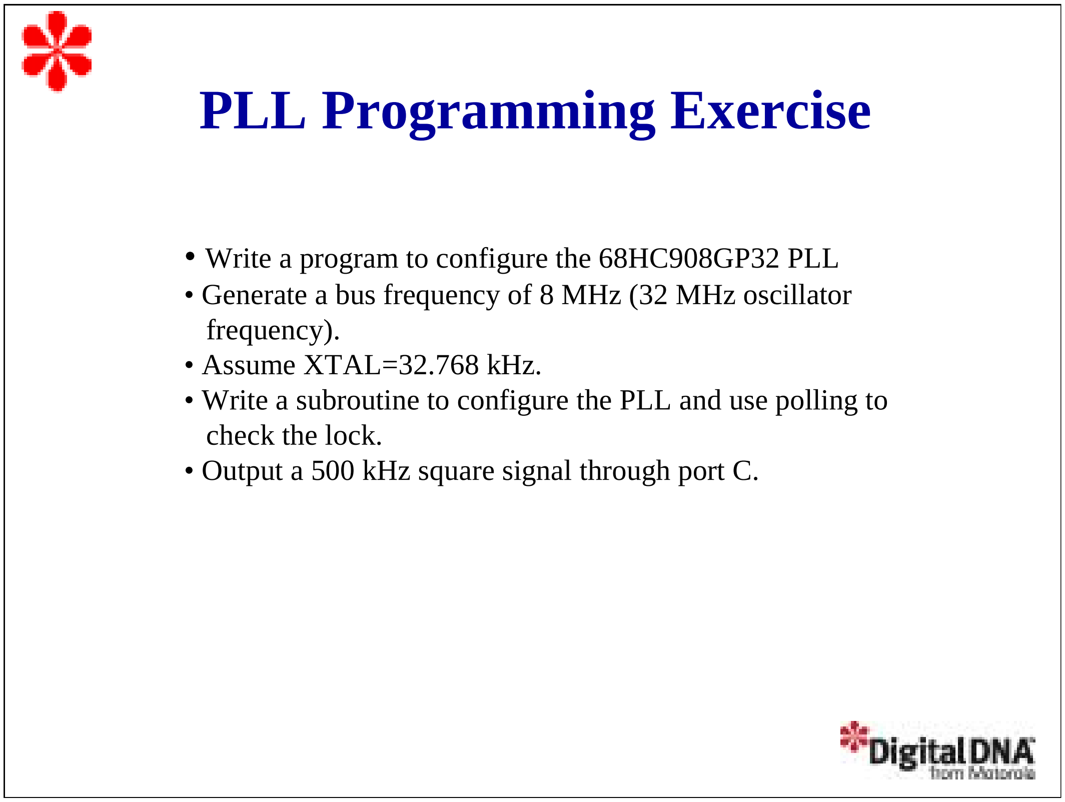
c) $\frac{1}{4}$ CGMXCLK

d) $\frac{1}{4}$ CGMVCLK

e) none of the above

What is the bus frequency when the BCS bit set to 1, the PLL is enabled, and the hardware center-of-range linear multiplier, L, set to a non-zero value? Click on your choice.

Answer: Setting the BCS bit to 1, enabling the PLL, and setting L to a non-zero value, selects the VCO frequency, CGMVCLK, as the source for the base clock, CGMOUT. In this case, CGMOUT is one-half CGMVCLK and the bus frequency is one-half CGMOUT. Therefore, the bus frequency is one-fourth CGMVCLK.



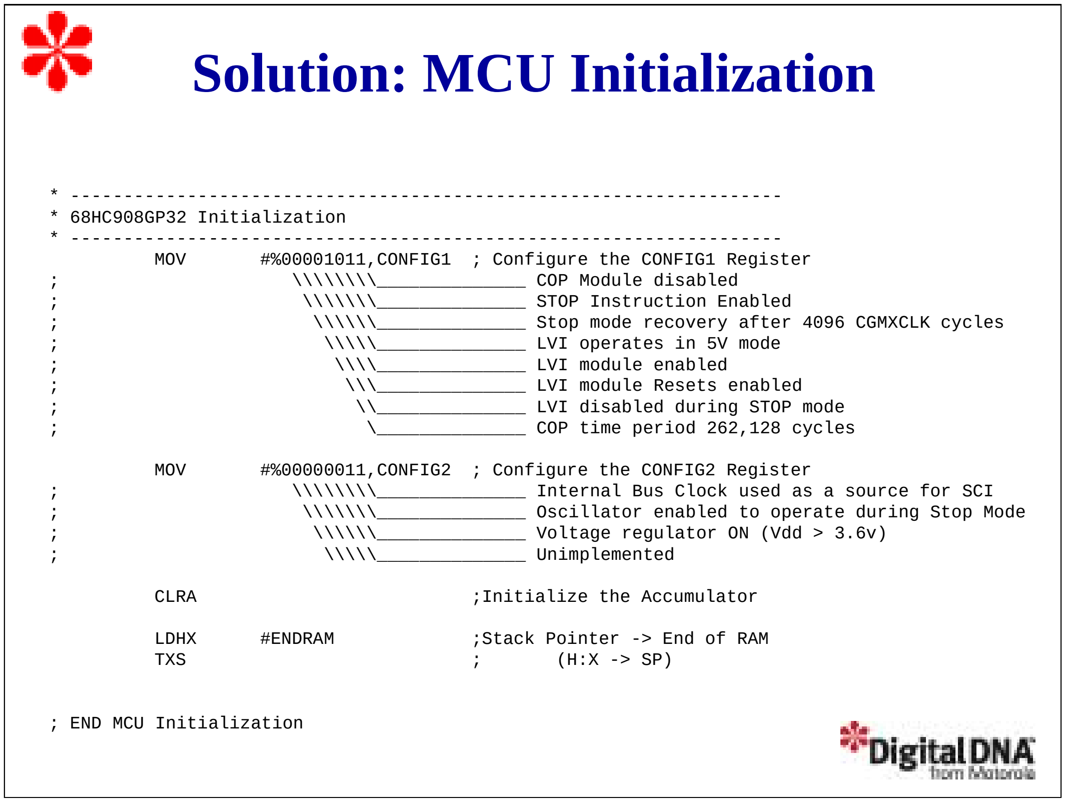
PLL Programming Exercise

- Write a program to configure the 68HC908GP32 PLL
- Generate a bus frequency of 8 MHz (32 MHz oscillator frequency).
- Assume XTAL=32.768 kHz.
- Write a subroutine to configure the PLL and use polling to check the lock.
- Output a 500 kHz square signal through port C.

Digital DNA from Motorola

Let’s complete this tutorial with a programming exercise. Write a program to configure the 68HC908GP32 PLL to generate an internal bus frequency of 8 MHz, or an oscillator frequency of 32 MHz, using an XTAL of 32.768 kHz. Write a subroutine to configure the PLL and poll the lock. To show that the internal bus frequency is actually running at 8MHz, output a 500 kHz square signal through the port C.

Take a moment to review the exercise instructions. When you are finished writing your program, click the Forward arrow to continue the tutorial and review the exercise solution.



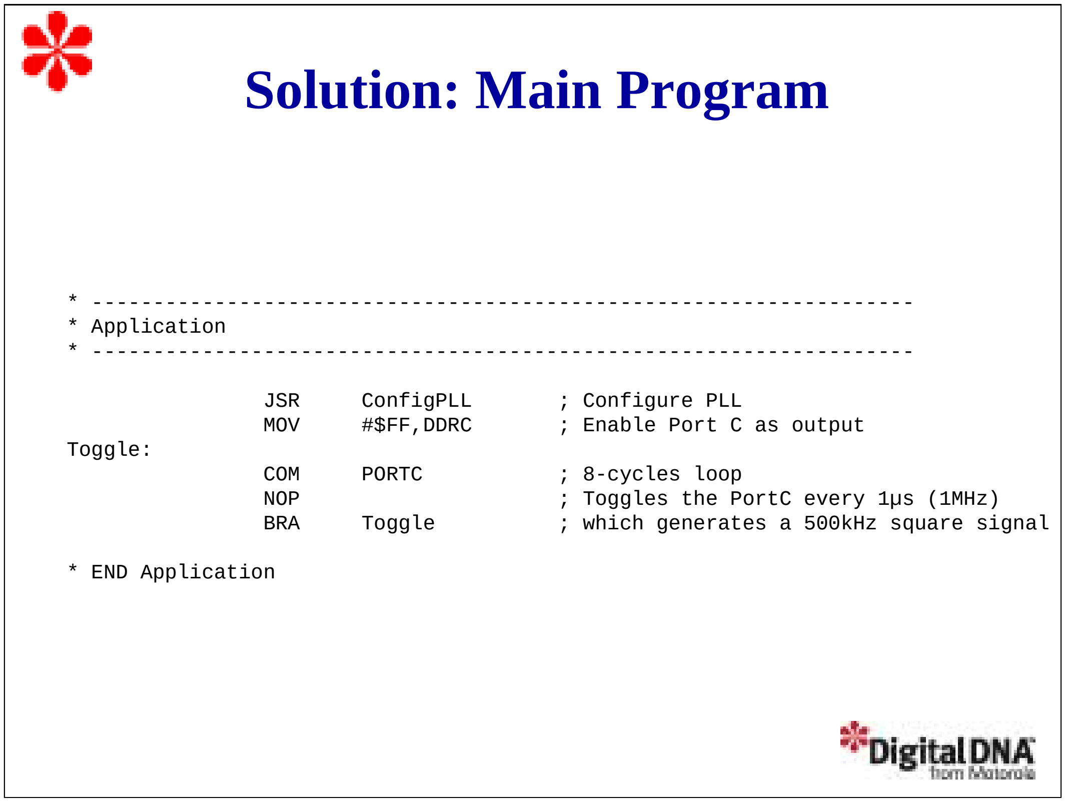
Solution: MCU Initialization

```
* -------------------------------------------------------------------
* 68HC908GP32 Initialization
* -------------------------------------------------------------------
          MOV       #%00001011,CONFIG1  ; Configure the CONFIG1 Register
;                      \\\\\\\\______________ COP Module disabled
;                       \\\\\\\______________ STOP Instruction Enabled
;                        \\\\\\______________ Stop mode recovery after 4096 CGMXCLK cycles
;                         \\\\\______________ LVI operates in 5V mode
;                          \\\\______________ LVI module enabled
;                           \\\______________ LVI module Resets enabled
;                            \\______________ LVI disabled during STOP mode
;                             \______________ COP time period 262,128 cycles

          MOV       #%00000011,CONFIG2  ; Configure the CONFIG2 Register
;                      \\\\\\\\______________ Internal Bus Clock used as a source for SCI
;                       \\\\\\\______________ Oscillator enabled to operate during Stop Mode
;                        \\\\\\______________ Voltage regulator ON (Vdd > 3.6v)
;                         \\\\\______________ Unimplemented

          CLRA                          ;Initialize the Accumulator

          LDHX      #ENDRAM             ;Stack Pointer -> End of RAM
          TXS                           ;       (H:X -> SP)


; END MCU Initialization
```

This section of code initializes the MCU. You can use this as a template for all of your programs. The first step is to initialize the MCU using the CONFIG1 and CONFIG2 registers. Remember, these registers can only be configured once after reset, so it is recommended that you do this first when starting.

In this example, we disable the COP in the CONFIG1 register so we don’t have to feed the counter to avoid a reset. We also enable the STOP instruction and enable the LVI to operate at 5v. Using CONFIG2, we select the internal bus clock as a source for the serial communication. We also enable the oscillator to operate during stop mode so we can use the periodic wake-up feature. Note that the configuration of the CONFIG2 register is not critical for this application and we could have used the default initialization.

The next instruction, CLRA, initializes the accumulator to avoid warnings in the debugger. The next two instructions redirect the stack pointer to the end of the RAM. Remember that in order to keep compatibility with the HC05 family, the HC08 family automatically initializes the stack pointer to $00FF after reset.

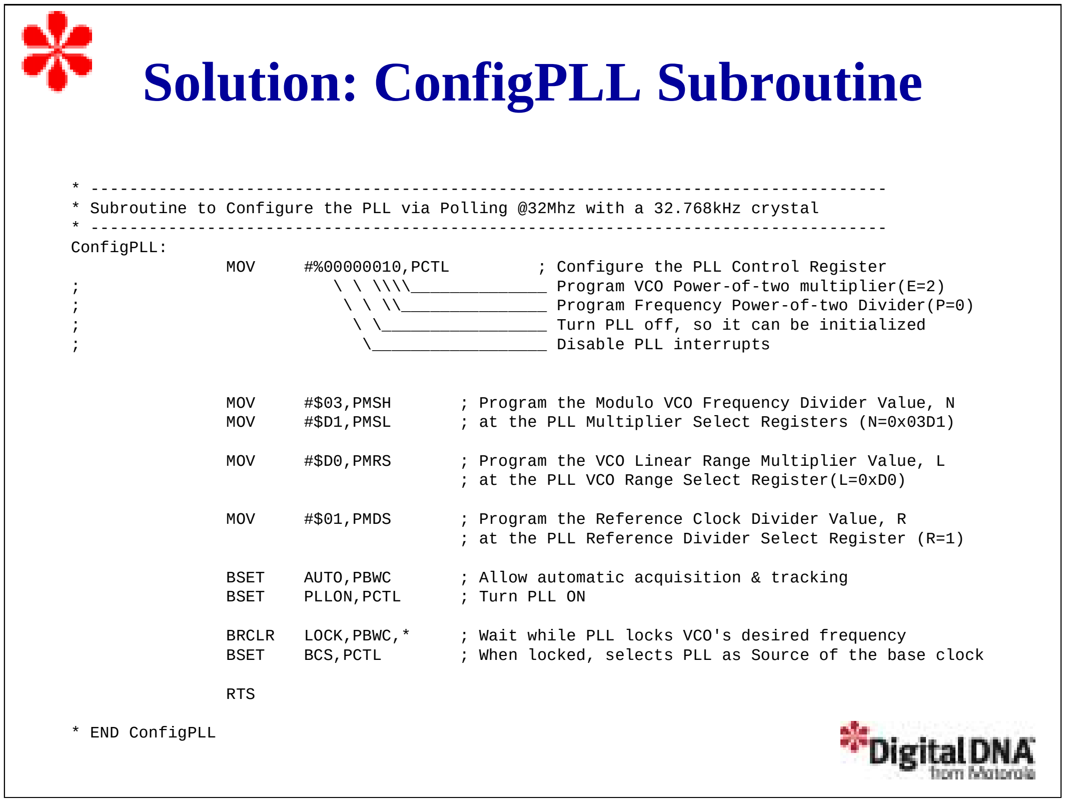


Solution: Main Program

```
* -------------------------------------------------------------------
* Application
* -------------------------------------------------------------------

                JSR     ConfigPLL       ; Configure PLL
                MOV     #$FF,DDRC       ; Enable Port C as output
Toggle:
                COM     PORTC           ; 8-cycles loop
                NOP                     ; Toggles the PortC every 1µs (1MHz)
                BRA     Toggle          ; which generates a 500kHz square signal

* END Application
```

This section of code is where we really start working with the PLL. The first thing we do is configure the PLL by jumping to a subroutine called ConfigPLL. We'll look at this subroutine in a moment.

In the main routine, we setup port C as the output. To generate a 500 kHz square wave, we need to toggle port C at a rate of 1 MHz, or once every microsecond. Running at an internal frequency of 8MHz (0.125us), we need an 8-cycle loop to generate a delay of 1us. COM PORTC is a 4-cycle instruction, NOP is a 1-cycle instruction, and BRA takes 3 cycles, for a total of 8 cycles. The routine toggles PORTC with the complement instruction (COM) every 8 cycles.



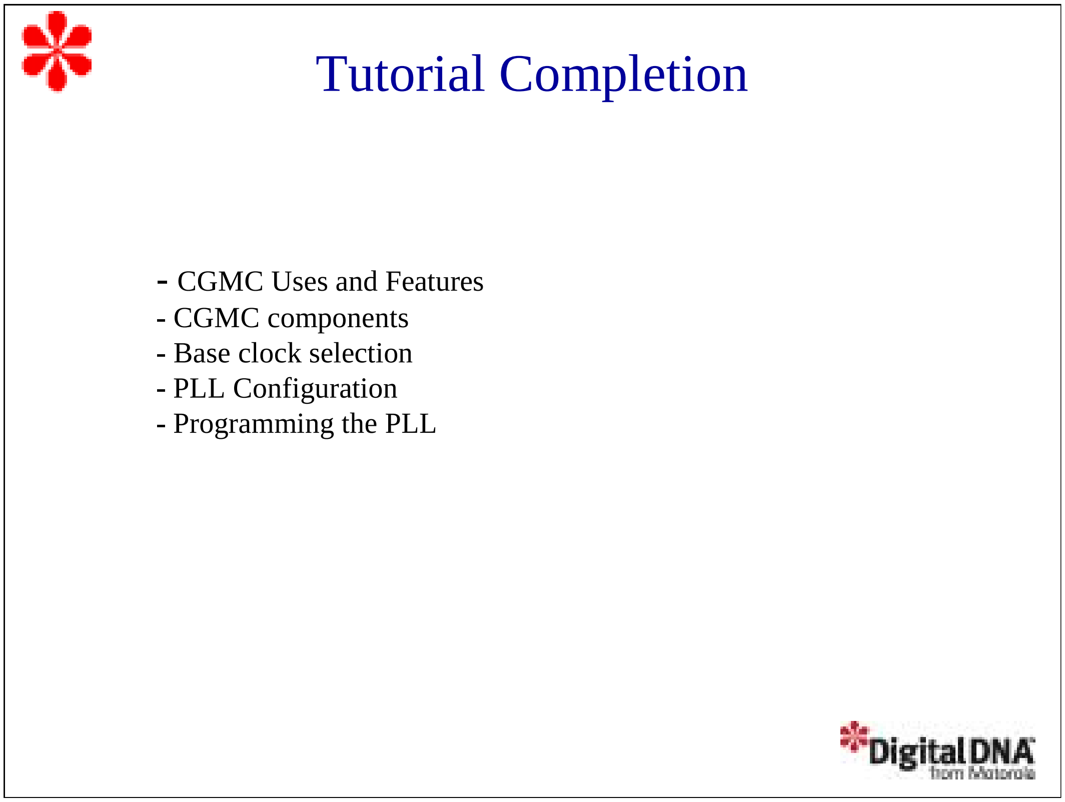
Solution: ConfigPLL Subroutine

```
* ----------------------------------------------------------------------------------
* Subroutine to Configure the PLL via Polling @32Mhz with a 32.768kHz crystal
* ----------------------------------------------------------------------------------
ConfigPLL:
                MOV     #%00000010,PCTL         ; Configure the PLL Control Register
;                          \ \ \\\\______________ Program VCO Power-of-two multiplier(E=2)
;                           \ \ \\_______________ Program Frequency Power-of-two Divider(P=0)
;                            \ \_________________ Turn PLL off, so it can be initialized
;                             \__________________ Disable PLL interrupts


                MOV     #$03,PMSH       ; Program the Modulo VCO Frequency Divider Value, N
                MOV     #$D1,PMSL       ; at the PLL Multiplier Select Registers (N=0x03D1)

                MOV     #$D0,PMRS       ; Program the VCO Linear Range Multiplier Value, L
                                        ; at the PLL VCO Range Select Register(L=0xD0)

                MOV     #$01,PMDS       ; Program the Reference Clock Divider Value, R
                                        ; at the PLL Reference Divider Select Register (R=1)

                BSET    AUTO,PBWC       ; Allow automatic acquisition & tracking
                BSET    PLLON,PCTL      ; Turn PLL ON

                BRCLR   LOCK,PBWC,*     ; Wait while PLL locks VCO's desired frequency
                BSET    BCS,PCTL        ; When locked, selects PLL as Source of the base clock

                RTS

* END ConfigPLL
```

Next, let’s look at the ConfigPLL subroutine. We use this subroutine to configure the PLL to generate a frequency of oscillation of 32MHz, or a divided-by-four bus frequency of 8MHz. We’ll use the values that we calculated in an earlier example, with R = 1, N = 0x03D1, P = 0, E = 2, and L = 0xD0.

First, we setup the PCTL register where we program E=2 and P=0. Also, we turn off the PLL so it can be initialized, and we disable PLL interrupts since we’ll use polling to check the lock. Next, program the value of N in the PLL multiplier select Registers and L in the PMRS register. Program the reference clock divider, R, in the PMDS register.

Next, we set the AUTO bit in the PLL bandwidth control register (PCWC) to allow automatic switching between acquisition and tracking modes. This is the recommended mode for most applications. Finally, we turn the PLL on and wait while the PLL locks the desired VCO frequency, which in this case is 32MHz. As soon as the PLL has locked the frequency, we must select the PLL as the source of the base clock.



Tutorial Completion

- CGMC Uses and Features
- CGMC components
- Base clock selection
- PLL Configuration
- Programming the PLL

In this tutorial, you’ve had an opportunity to work with the 68HC08 CGMC. You’ve learned about the features of the main CGMC components: the crystal oscillator circuit, the PLL, and the base clock selector circuit. You’ve also learned how to program the PLL for a specific bus frequency using the CGMC registers, and demonstrated this by writing a program to generate a 500 kHz square signal.