DCO CALIBRATION OF MSP430F2618 ULTRA LOW POWER MICROCONTROLLER

[1]Divya, [2]Swati Chodhaury

[1,2]EC, Raj Kumar Goel Institute of Engineering and Technology for Women Ghaziabad/MTU, (India)

ABSTRACT

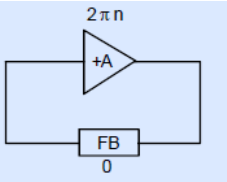
MSP430F2618 is a 16 bit microcontroller based on RISC system designed for ultralow power consumption and are manufactured by TEXAS INSTRUMENTS. DCO is the most important component of MSP430F2618 which is to be calibrated in order to control its oscillation frequency and ascertain its operational integrity. Oscillation frequency of DCO is controlled using software (Code composite studio). We also studied that DCO can be calibrated by tuning it to a more accurate external source.

I.INTRODUCTION

The MSP430 family of ultralow-power microcontrollers consists of several devices featuring different sets of peripherals targeted for various applications. The architecture, combined with five low-power modes is optimized to achieve extended battery life in portable measurement applications. The device features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that contribute to maximum code efficiency. The DCO which is calibrated allows wake-up from low-power modes to active mode in less than 1 µs.

The MSP430F261x and MSP430F241x series are microcontroller configurations with two built-in 16-bit timers, a fast 12-bit A/D converter, a comparator, dual 12-bit D/A converters, four universal serial communication interface (USCI) modules, DMA, and up to 64 I/O pins. The MSP430F241x devices are identical to the MSP430F261x devices, with the exception that the DAC12 and the DMA modules are not implemented.

Sensor systems, industrial control applications, wireless communication system and hand-held meters are major applications. The 12mmx12mm LQFP-64 package is also available as a non-magnetic package for application of Medical imaging. Our prime focus here is digitally controlled oscillator (DCO) ,how it's used as source and why need calibration. It is necessary to calibrate DCOs before use to ensure their operational integrity and to know the uncertainty of the frequencies of operation. Any oscillator, be it RC, LC or Crystal controlled, requires two basic conditions to be met for it to operate at the desired frequency; its loop gain must be greater than unity and its loop phase shift must be zero at that frequency (Barkhausen stability criterion).



Modern system(Microcontroller based DCOs) usually use some form of digital frequency synthesis to generate their VFO signal. Smaller designs, lack of moving parts, and the ease with which present frequencies can be stored and manipulated in the digital computer that is usually embedded in the design for other purposes are major advantages. Digital frequency synthesis relies on stable crystal controlled reference frequency sources. Crystal controlled oscillators are more stable than inductively and capacitively controlled oscillators (VCOs). They tend to be more stable, more repeatable, have fewer and lower harmonics and lower noise than all the alternatives in their cost-band.

II. MSP430 AND CLOCKING SYSTEM

MSP430 series of microcontroller has a flexible clocking system. Flexible clocking system helps this microcontroller achieve it's purpose of ultra low power application. The MSP430 MCU clock system has the ability to enable and disable various clocks and oscillators which allow the device to enter various low-power modes (LPMs). The flexible clocking system optimizes overall current consumption by only enabling the required clocks when

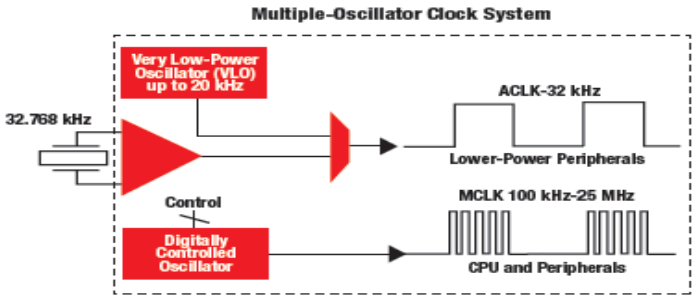


Fig-1.1 Multiple-Oscillator Clock System

Main Clock (MCLK) – CPU source that may be driven by the internal
Digitally Controlled Oscillator (DCO) up to 25 MHz or with external crystal.

Auxiliary Clock (ACLK) – Source for individual peripheral modules driven by the internal low-power oscillator or external crystal.

Sub-Main Clock (SMCLK) – Source for faster individual peripheral
modules that may be driven by the internal DCO up to 25 MHz or with external crystal.

For the above clock signals the basic clock module+ msp430series microcontroller includes two, three or four clock sources

LFXT1CLK: Low-frequency/high-frequency oscillator that can be used with low-frequency watch crystals or external clock sources of 32768 Hz or with standard crystals, resonators, or external clock sources in the 400-kHz to 16-MHz range.

XT2CLK: Optional high-frequency oscillator that can be used with standard crystals, resonators, or external clock sources in the 400-kHz to 16-MHz range.

DCOCLK: Internal digitally controlled oscillator (DCO).

VLOCLK: Internal very low power, low frequency oscillator with 12-kHz typical frequency

III. NEED FOR CALIBRATION

Running an MSP430 at lower voltages changes the frequency of the internal DCO. High frequency external crystals have their own entire set of problems. Clock settings between chips are not transferable and setting the DCO and BCS values for one chip doesn't not result in the same frequency for another. Hence, every chip needs to be calibrated to run at a custom frequency before running the actual code you're trying to deploy.

IV. DCO CALIBRATION

DCO is the major component of MSP430. The frequency at which the DCO oscillates can be adjusted by setting the DCO registers. In this way, the DCO can be tuned by incrementally changing the registers and comparing the resultant frequency against a known frequency. When the speed of a slower clock source is known, such as a 32-kHz watch crystal, the DCO speed can be adjusted until a specific number of DCO cycles occur in one cycle of the slower clock.

To accomplish this, a Capture/Compare Register of Timer_A is initialized in Capture mode. In this mode, it captures the value of Timer_A when a low-to-high transition occurs on an internal signal. In this case, the internal signal is ACLK. When SMCLK is driving the timer, the captured value becomes the number of SMCLK cycles since the last ACLK low-to-high transition. When the number of SMCLK cycles are known, the DCO can be adjusted and measured again using same capture method previously outlined. In this way, the DCO can be tuned to a specific multiple of the known ACLK frequency. For increased accuracy, the ACLK signal can be divided by eight to increase granularity. This is known as dco calibration i.e. tuning it's frequency to known much accurate clock source (here watch crystal is used).

V.CONFIGURING THE DCO

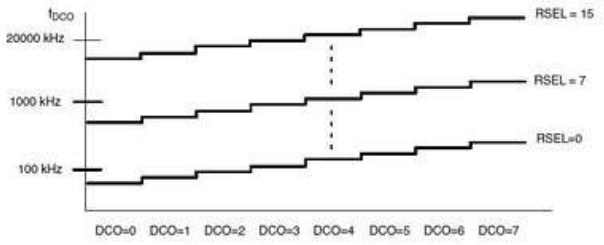


Fig-2.1 How to Select The different Possible Frequencies Of DCO

The MSP430's Digitally Controlled Oscillator is one of the most important peripherals to be able to manage. As it requires no external parts and is controlled completely by software it's the easiest of the clock sources to configure. The DCO (within the BCS+) is configured with two registers: DCOCTL and BCSCTL1. Look up the maps of these two registers in the x2xx Family User's Guide, page 5-14.This image comes directly from the User's Guide. It explains how to select the different possible frequencies for the DCO. The total range of frequencies is divided into 16 overlapping ranges. So Range 7's low end overlaps Range 6's high end, and Range 7's high end overlaps Range 8's low end. Make sense? Ok, each range is further subdivided into 8 possible Ok, each range is further subdivided into 8 possible frequencies. If you plot all of these out, you get something similar to the above image, but with 16 lines instead of the three representative lines shown here. There are then 128 distinct frequencies available to the DCO, with some overlap.

VI. USING THE DCO LIBRARY

It is often necessary to have the digitally controlled oscillator (DCO) of an MSP430 tuned to a specific frequency. Simply setting the DCO register values to a constant value across all devices is not a method that can ensure accurate results due to device variances. DCO must be tuned based on a known frequency in order to have accuracy. This is done automatically using the FLL In 4xx devices. The purpose of the FLL is to keep the DCO running at a certain multiple of ACLK. In 1xx and 2xx devices, which do not have the FLL, a similar tuning process can be accomplished using software and timers.

VII. SAFETY FACTORS

Several safety measures have been added to the library in order to ensure reliable and safe use of the DCO library.
In order to prevent the library from becoming trapped in an infinite loop, a maximum number of 10,000loops are allowed. If the desired frequency is not set after 10,000 iterations, the function returns 0xFF, in order to signify the timeout has occurred. It should be noted that this timeout is loop count based and not time based. This means that a timeout does not occur if no ACLK signal is present. On 2xx devices, the DCO speed should not exceed the specified 16 MHz. As an additional safety factor,
2xx devices will not increment the DCO settings above the value given in the factory-calibrated 16-MHz register settings. If this value is reached, the function exits, returning 0□02 to indicate the DCO is set to the fastest setting.
In order to ensure a more consistent time for setting the DCO, a tolerance has been built into the set DCO routine. The routine exits if the measured DCO is exactly set to the desired multiplier or if the DCO speed shown.

Below is the basic program this program is a demonstration of setting the DCO to 2 MHz by tuning it to a 32-kHz watch crystal.

After a delay to ensure the startup of the 32-kHz crystal, the function set DCO is called. This function takes one parameter, an integer describing the desired frequency of the DCO. In order to tune the DCO, this routine divides ACLK by eight, then increases or decreases the DCO settings until the number of DCO cycles per ACLK cycle is equal to the number passed to the set DCO routine.
Put another way, the resultant DCO frequency is:

DCO = parameter □□(32768/8)

```
#include <msp430x11x1.h>

#include "DCO Library.h"

void main(void)

{ volatile unsigned int I;

int result;

WDTCTL = WDTPW +WDTHOLD; // Stop Watchdog Timer

P1DIR |= 0x12; // P1.1 and P1.4 outputs

P1SEL |= 0x10; // P1.4 SMCLK output

P2DIR |= 0x01; // P2.0 output

P2SEL |= 0x01; // P2.0 ACLK output

for( I = 0; I < 0xFFFF; I++){} // delay for ACLK startup

result =TI_setDCO (TI_setDCO_2MHZ);

if( result == TI_DCO_SET_TO_SLOWEST ) // returned result if DCO registers hit min

{

while(1); // trap the CPU if hit

}

else if( result ==TI_ DCO_SET_TO_FASTEST ) // returned result if DCO registers hit max

{

while(1); // trap the CPU if hit

}

while(1); // trap the CPU if hit

}

else if( result == TI_DCO_TIMEOUT_ERROR ) // result if DCO takes >10000 loops
```

```
{

while(1); // trap the CPU if hit

}

while(1)

{

P1OUT |= 0x02; // P1.1 = 1

P1OUT &= ~0x02; // P1.1 = 0

}

}
```

The required dco frequency can be set by tuning it to watch crystal .In another direct and interesting way dco can be tuned to specified frequency by using calibration registers calb1_--HZ and caldco—HZ located in the flash memory provided by TI. directly using it to get desired DCO frequency.

Below is the program to do so:

```
//*******************************************************************************
//  MSP430x26x Demo - Basic Clock, Output Buffered SMCLK, ACLK and MCLK/10
//
//  Description: Buffer ACLK on P5.6, SMCLK(DCO) on P5.5, MCLK on P5.4 and
//  MCLK/10 on P5.3.
//  ACLK = LFXT1 = 32768Hz, MCLK = SMCLK = CALxxx_8MHZ = 8MHz
//  //* External watch crystal on XIN XOUT is required for ACLK *//
//
//           MSP430F261x/241x
//         -----------------
//     /|\|              XIN|-
//      | |                 | 32kHz
//      --|RST          XOUT|-
//        |                 |
//        |            P5.6|-->ACLK = 32kHz
//        |            P5.5|-->SMCLK = 8MHz
//        |            P5.4|-->MCLK = DCO
//        |            P5.3|-->MCLK/10
//
//  B. Nisarga
//  Texas Instruments Inc.
//  September 2007
//  Built with CCE Version: 3.2.0 and IAR Embedded Workbench Version: 3.42A

//  B. Nisarga
//  Texas Instruments Inc.
//  September 2007
//  Built with CCE Version: 3.2.0 and IAR Embedded Workbench Version: 3.42A
```

```
//*****************************************************************************
#include"msp430f2618.h"

int main(void)
{
WDTCTL = WDTPW + WDTHOLD;                 // Stop Watchdog Timer  if (CALBC1_8MHZ==0xFF)
                                  // If calibration constant erased
int i;
BCSCTL1 = CALBC1_1MHZ;
DCOCTL = CALDCO_1MHZ;
BCSCTL3|= XT2S1;
for (i = 0xFF; i > 0; i--);             // Time for flag to set
BCSCTL1 |= XTS;                         // ACLK = LFXT1 = HF XTAL
BCSCTL2 |= SELM_2 + SELS;                 // MCLK/SMCLK = LFXT1 (safe)
BCSCTL3 |= XT2S_2;                      // 16MHz crystal
BCSCTL3 &= ~XCAP0;
do
{
IFG1 &= ~OFIFG;                         // Clear OSCFault flag
for ( = 0xFF; i > 0; i--);              // Time for flag to set
}
while (IFG1 & OFIFG);                     // OSCFault flag still set?
P5DIR |= 0x70;                          // P5.4/5/6= output directionP5DIR |= 0x70;                     // P5.4/5/6= output
direction
P5SEL |= 0x70;                          // P5.4/5/6= MCLK/SMCLK/ACLK option select
while(1);
}
```

but these calibration constants are sometimes erased or changed due to different programming and burning design performed on the MCU .so these needs to be replaced with correct constant values. For this purpose a program is used REPLACES THE TI FACTORY-PROGRAMMED DCO CALIBRATION CONSTANTS LOCATED IN INFO segment A WITH NEW VALUES.

```
//*****************************************************************************
// Custom DCO settings- based on TI code example
// MSP430F20xx Demo - DCO Calibration Constants Programmer
//
// NOTE: THIS CODE REPLACES THE TI FACTORY-PROGRAMMED DCO CALIBRATION
// CONSTANTS LOCATED IN INFOA WITH NEW VALUES. USE ONLY IF THE ORIGINAL
// CONSTANTS ACCIDENTALLY GOT CORRUPTED OR ERASED.
//
//
// MSP430F20xx
// ---------------
// /|\| XIN|-
// | | | 32kHz
// --|RST XOUT|-
// | |
// | P1.0|--> LED
// | P1.4|--> SMLCK = target DC
// Orignal Code By
// A. Dannenberg
// Texas Instruments Inc.
// May 2007
// Built with CCE Version: 3.2.0 and IAR Embedded Workbench Version: 3.42A
//*****************************************************************************
```

```
//*******************************************************************************
/*Flash Custom DCO settings, This will replace The default 1MHz */
/* to use
//Custom calibration
BCSCTL1 = CALBC1_1MHZ; // Set range
DCOCTL = CALDCO_1MHZ; // Set DCO step + modulation
*/
//ACLK = LFXT1/8 = 32768/8, MCLK = SMCLK = target DCO
//* External watch crystal installed on XIN XOUT is required for ACLK *//
//*******************************************************************************
#include <msp430f2618.h>
#include "DCO_Library.h"
unsigned char CAL_DATA[10];
int j;
char *Flash_ptrA;
void main(void)
{
volatile unsigned int I;
WDTCTL = WDTPW +WDTHOLD; // Stop Watchdog Timer
unsigned char CAL_DATA[10];
int j;
char *Flash_ptrA;
void main(void)
{
volatile unsigned int I;
WDTCTL = WDTPW +WDTHOLD; // Stop Watchdog Timer
P1DIR |= 0x12; // P1.1 and P1.4 outputs
P1SEL |= 0x10; // P1.4 SMCLK output
P2DIR |= 0x01; // P2.0 output
P2SEL |= 0x01; // P2.0 ACLK output
j=0;
for( I = 0; I < 0xFFFF; I++){} // delay for ACLK startup
TI_SetDCO(TI_DCO_12MHZ); // Set DCO and obtain constants
CAL_DATA[j++] = DCOCTL;
CAL_DATA[j++] = BCSCTL1;
TI_SetDCO(TI_DCO_12MHZ); // Set DCO and obtain constants
CAL_DATA[j++] = DCOCTL;
CAL_DATA[j++] = BCSCTL1;
TI_SetDCO(TI_DCO_8MHZ); // Set DCO and obtain constants
CAL_DATA[j++] = DCOCTL;
CAL_DATA[j++] = BCSCTL1;
TI_SetDCO(TI_DCO_1MHZ); // Set DCO and obtain constants
CAL_DATA[j++] = DCOCTL;
CAL_DATA[j++] = BCSCTL1;
P1OUT |= 0x02; // P1.1 = 1
P1OUT &= ~0x02; // P1.1 = 0
Flash_ptrA = (char *)0x10C0; // Point to beginning of seg A
FCTL2 = FWKEY + FSSEL0 + FN1; // MCLK/3 for Flash Timing Generator
FCTL1 = FWKEY + ERASE; // Set Erase bit
FCTL3 = FWKEY + LOCKA; // Clear LOCK & LOCKA bits
P1OUT |= 0x02; // P1.1 = 1
P1OUT &= ~0x02; // P1.1 = 0
Flash_ptrA = (char *)0x10C0; // Point to beginning of seg A
FCTL2 = FWKEY + FSSEL0 + FN1; // MCLK/3 for Flash Timing Generator
FCTL1 = FWKEY + ERASE; // Set Erase bit
FCTL3 = FWKEY + LOCKA; // Clear LOCK & LOCKA bits
```

```
*Flash_ptrA = 0x00; // Dummy write to erase Flash seg A
FCTL1 = FWKEY + WRT; // Set WRT bit for write operation
Flash_ptrA = (char *)0x10F8; // Point to beginning of cal consts
for (j = 0; j < 10; j++)
*Flash_ptrA++ = CAL_DATA[j]; // re-flash DCO calibration data
FCTL1 = FWKEY; // Clear WRT bit
FCTL3 = FWKEY + LOCKA + LOCK; // Set LOCK & LOCKA bit
P1OUT |= 0x02; // P1.1 = 1
P1OUT &= ~0x02; // P1.1 = 0}
```

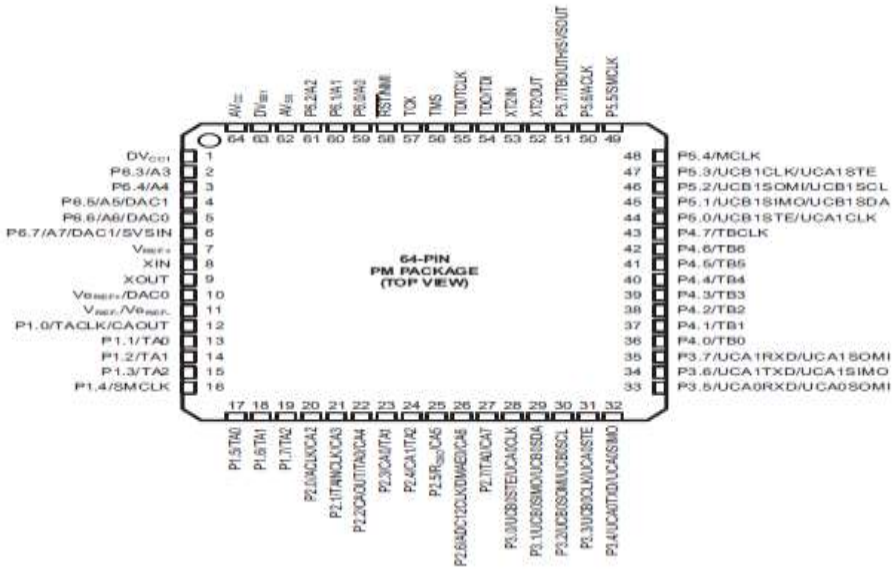
These calibrated constants can be used any time in future in any application.

VIII. DETAILS OF READINGS

Five microcontrollers were taken of the series msp430f2618 which were provided.

Their readings i.e. ACLK, MCLK and SMLK readings were taken directly from the MCU Pin no 50, 48, 49 respectively using DSO. The block diagram showing pin configuration MSP430f2618 ultra low MCU is shown below.

Device Pinout, MSP430F261x, 64-Pin PM Package



MSP4302618	freq	pin	BEFORE CALIBRATION	AFTER CALIBRATION
MCU-1	1MHZ	P5.4/MCLK(48)	15.0582MHZ	998.822KHZ
		P5.5/SMCLK(49)	15.0635MHZ	998.915KHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	8MHZ	P5.4/MCLK(48)	15.0542MHZ	7.98907MHZ
		P5.5/SMCLK(49)	15.0625MHZ	7.99075MHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	12MHZ	P5.4/MCLK(48)	15.0462MHZ	11.9979MHZ

		P5.5/SMCLK(49)	15.0185MHZ	12.0006MHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	16MHZ	P5.4/MCLK(48)	15.0902MHZ	16.0053MHZ
		P5.5/SMCLK(49)	15.0225MHZ	16.0066MHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	1MHZ	P5.4/MCLK(48)	15.0152MHZ	1.00189MHZ
MCU-2		P5.5/SMCLK(49)	15.0565MHZ	1.00182MHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	8MHZ	P5.4/MCLK(48)	15.0190MHZ	7.98961MHZ
		P5.5/SMCLK(49)	15.0225MHZ	7.99075MHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	12MHZ	P5.4/MCLK(48)	15.0125MHZ	11.9944MHZ
		P5.5/SMCLK(49)	15.0490MHZ	11.9950MHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	16MHZ	P5.4/MCLK(48)	15.1908MHZ	16.0128MHZ
		P5.5/SMCLK(49)	15.0789MHZ	15.0098MHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	1MHZ	P5.4/MCLK(48)	15.0145MHZ	999.425KHZ
		P5.5/SMCLK(49)	15.0865MHZ	999.201KHZ
MCU-3		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	8MHZ	P5.4/MCLK(48)	15.0675MHZ	8.00209MHZ
		P5.5/SMCLK(49)	15.1225MHZ	8.00293MHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	12MHZ	P5.4/MCLK(48)	15.0789MHZ	11.8490MHZ
		P5.5/SMCLK(49)	15.0540MHZ	11.9480MHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	16MHZ	P5.4/MCLK(48)	15.1908MHZ	15.9899MHZ
		P5.5/SMCLK(49)	15.0129MHZ	15.9947MHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ
	1MHZ	P5.4/MCLK(48)	15.0890MHZ	999.435kHZ
		P5.5/SMCLK(49)	15.0345MHZ	999.438kHZ
		P5.6/ACLK(50)	32.7677KHZ	32.7677KHZ

MCU-4	8MHZ	P5.4/MCLK(48) P5.5/SMCLK(49) P5.6/ACLK(50)	15.0108MHZ 15.0998MHZ 32.7677KHZ	7.98947MHZ 7.99267MHZ 32.7677KHZ
	12MHZ	P5.4/MCLK(48) P5.5/SMCLK(49) P5.6/ACLK(50)	15.0890MHZ 15.0760MHZ 32.7677KHZ	12.0044MHZ 32.7677KHZ
	16MHZ	P5.4/MCLK(48) P5.5/SMCLK(49) P5.6/ACLK(50)	15.0709MHZ 15.0128MHZ 32.7677KHZ	15.9772MHZ 15.9831MHZ 32.7677KHZ
	1MHZ	P5.4/MCLK(48) P5.5/SMCLK(49) P5.6/ACLK(50)	15.1345MHZ 15.0975MHZ 32.7677KHZ	1.00054MHZ 1.00067MHZ 32.7677KHZ
MCU-5	8MHZ	P5.4/MCLK(48) P5.5/SMCLK(49) P5.6/ACLK(50)	15.1208MHZ 15.0398MHZ 32.7677KHZ	7.99483MHZ 7.99765MHZ 32.7677KHZ
	12MHZ	P5.4/MCLK(48) P5.5/SMCLK(49) P5.6/ACLK(50)	15.0390MHZ 15.0260MHZ 32.7677KHZ	11.9982MHZ 11.9972MHZ 32.7677KHZ
	16MHZ	P5.4/MCLK(48) P5.5/SMCLK(49) P5.6/ACLK(50)	15.2609MHZ 15.1268MHZ 32.7677KHZ	15.9906MHZ 15.9924MHZ 32.7677KHZ

IX. CONCLUSION

We have studied the different clock signals along with their sources .specially DCO(digitally controlled oscillator) how it's oscillation frequency can be controlled using software .we also studied different ways to calibrate DCO by tuning it to a more accurate external source. Difference between the clock readings before and after calibration were observed, recorded and Justified .there is small margin of inaccuracy within small range of few hertz which varied randomly while measuring clock readings.

REFRENCES

[1].MSP430 2618 User Manual.

[2]Application report – using the DCO library.

[3]MSP430F2618 Device Erratasheet

[4]www.ti.com

[5]http://mspsci.blogspot.in/2010/07/tutorial-08-b-configuring-dco.html

[6]http://justinstech.org/2011/05/msp430-custom-calibration-for-dco/

[7]http://www.embeddedrelated.com/groups/msp430/show/49720.php