**Assignment 4:  
Interrupts and Timers**

**Cal Poly CPE 329-01**

**Spring 2017**

Brendan Baronia

Melinda Ong

Professor Gerfen, Jeffrey B.

**April 21, 2017**

**Clock Calculation Arithmetic:**

**24 KHz 25% duty Clock:**

To implement the 25% duty cycle, we can leave LED on for 1 interrupt, off for 3 interrupts

**24 KHz 50% duty Square Wave:**

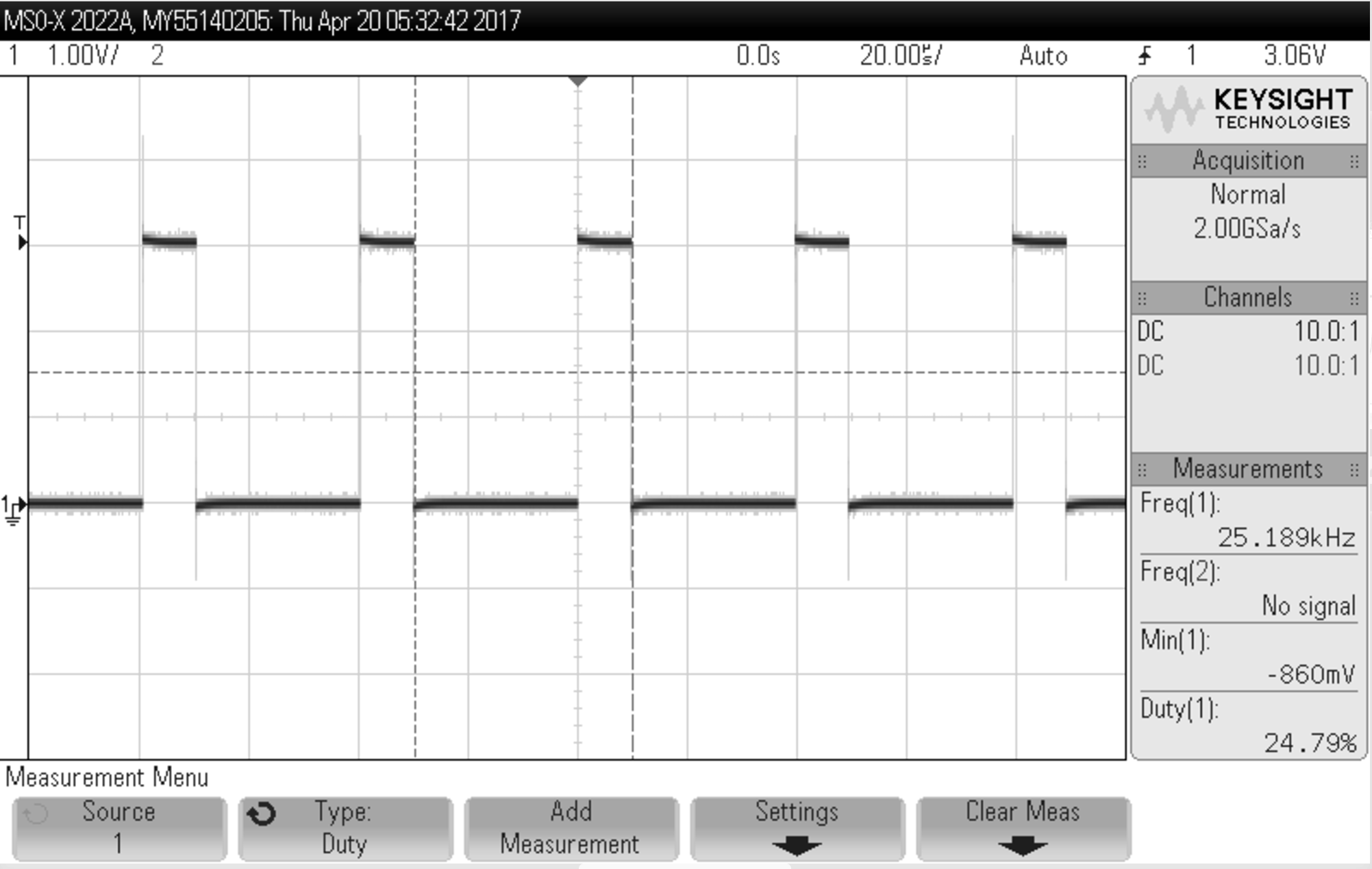
**20 second 50% duty Square Wave:**

By counting to 500 interrupts, we can implement a 20 second period without the counter rolling over.

**2 Bit Timer:**

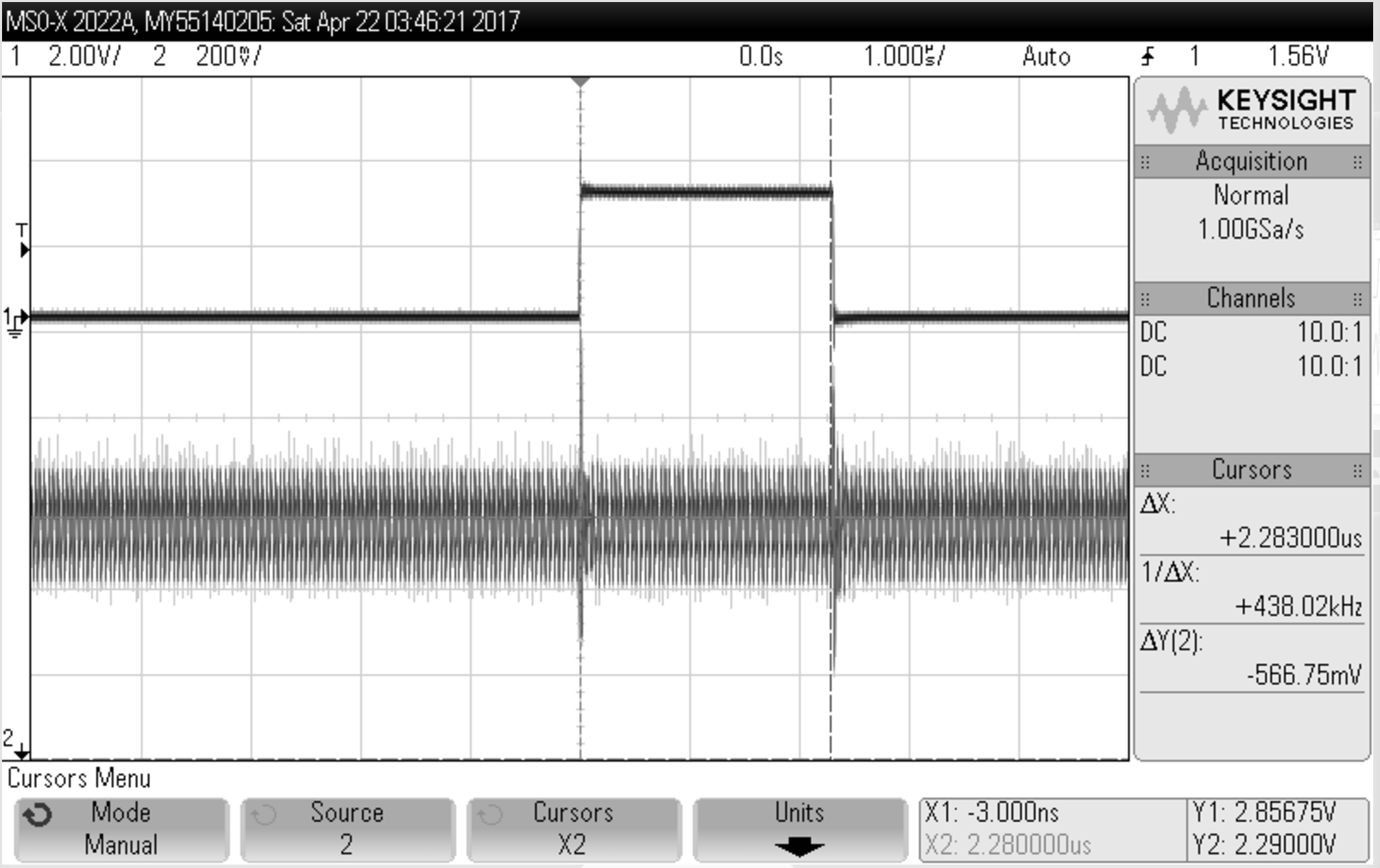
**Oscilloscope Screenshots:**

Step 2.) 25 KHz Clock with 25% Duty Cycle

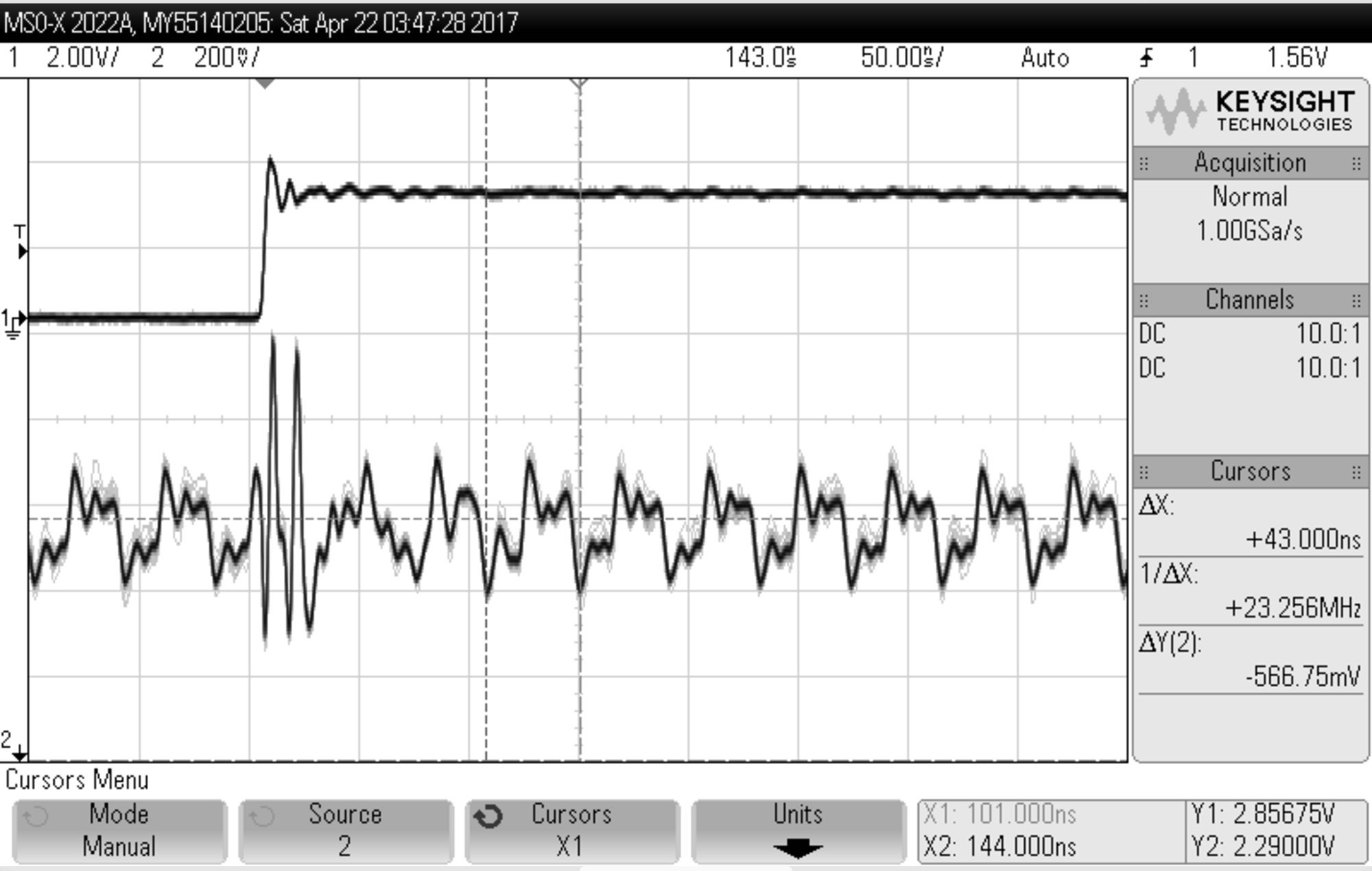
****

**Figure 1:** 25KHz clock (step 2) with a 25% duty cycle using DCO = 24MHz

Step 4.) ISR Processing Measurement



**Figure 2:** ISR execution and MCLK (step 4)

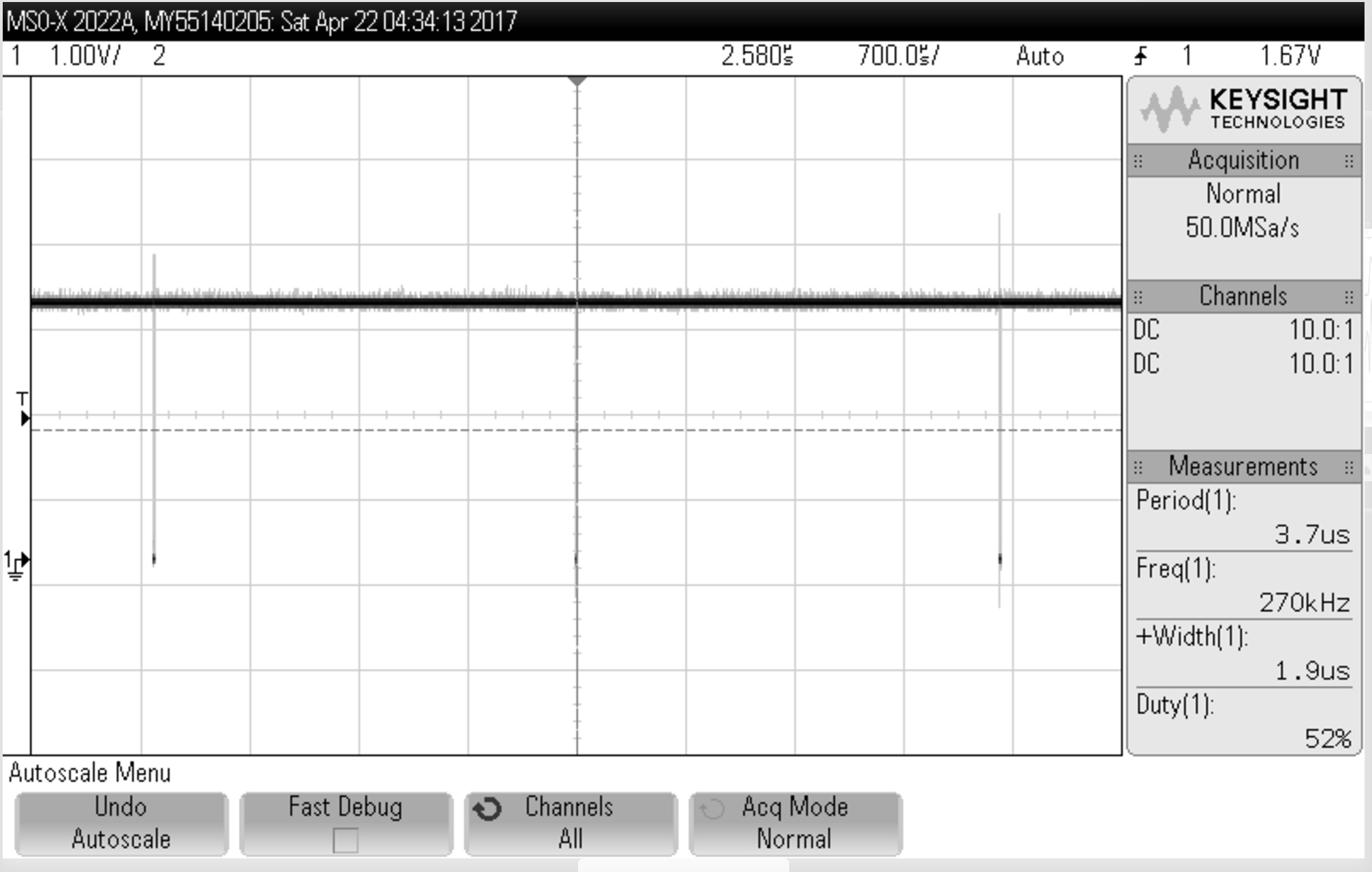


**Figure 3:** [Zoomed] ISR execution and MCLK (step 4)

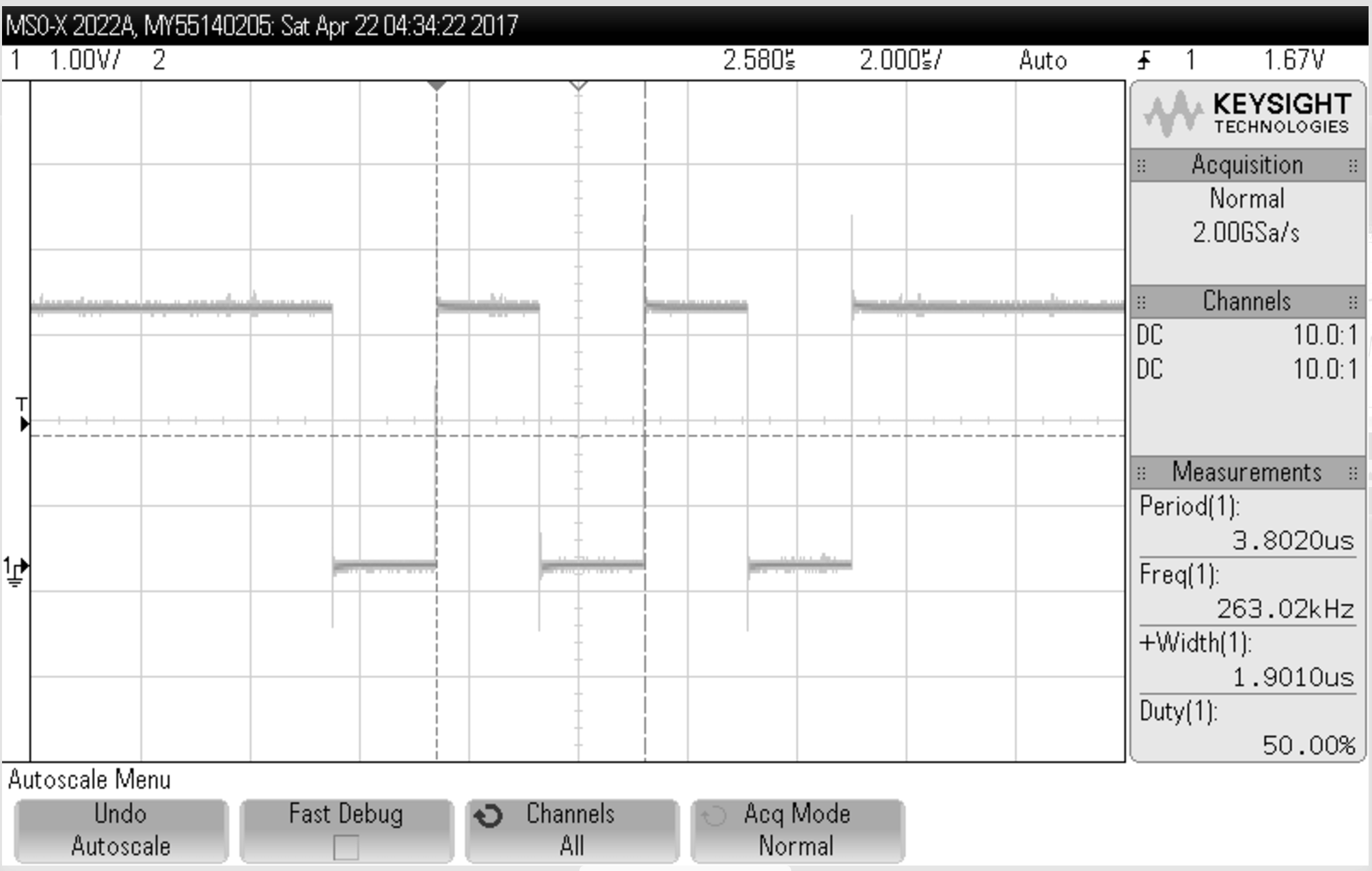
In **Figure 2**, a period of 2.283 microseconds is observed to execute an empty interrupt. In **Figure 3**, a period of 43.000 nanoseconds is observed to execute a single MCLK cycle.

Therefore, it takes approximately 53 MCLK cycles for ISR execution to measure a bit.

Step 5.) Shortest Pulse and Failing ISR

****

**Figure 4:** Shortest pulse (step 5)

****

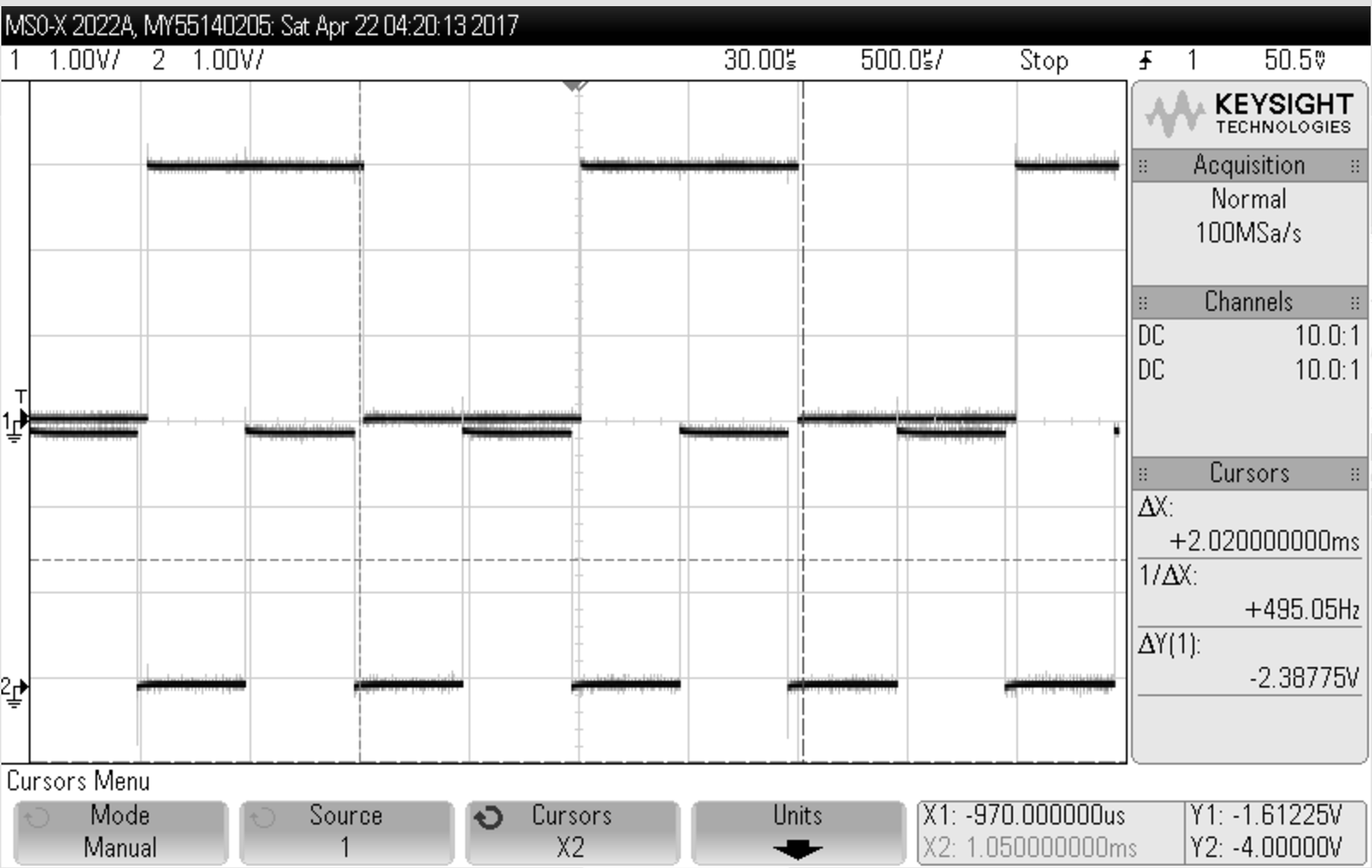
**Figure 5:** [Zoomed] Shortest pulse (step 5)

In **Figure 4**, a period of 3.8020 microseconds and frequency of 263.02 kHz is observed for the shortest pulse.

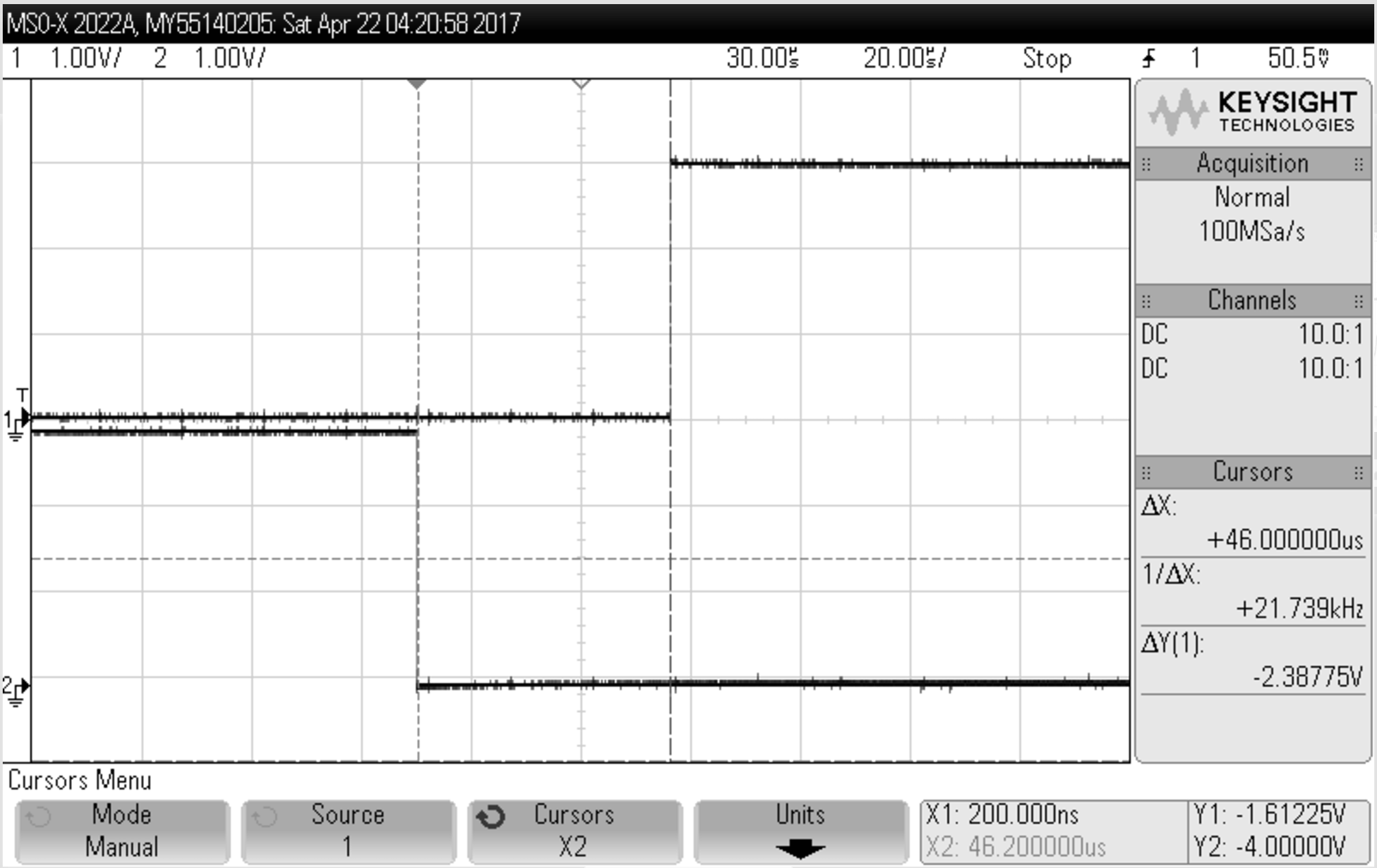
Step 6.) Clock with 50% Duty Cycle and 20-Second Period

**Youtube video of 20-second clock:** <https://youtu.be/20bfWFgdquA>

Step 8.) 2-Bit Counter



**Figure 6:** 2-bit counter (step 8)



**Figure 7:** [Zoomed] 2-bit counter (step 8)

In **Figure 7**, a delay of 46 microseconds is observed between Timer\_A0 CCR0 and Timer\_A0 CCR1.

**Code:**

***i.) main.c with ISR for 25 KHz clock system***

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// MSP432P401 Demo - Timer0\_A3, Toggle P1.0, CCR0 Cont Mode ISR, DCO SMCLK  
//  
// Description: Toggle P1.0 using software and TA\_0 ISR. Timer0\_A is  
// configured for continuous mode, thus the timer overflows when TAR counts  
// to CCR0. In this example, CCR0 is loaded with 50000.  
// ACLK = n/a, MCLK = SMCLK = TACLK = default DCO = ~3MHz  
//  
// MSP432P401x  
// ---------------  
// /|\| |  
// | | |  
// --|RST |  
// | |  
// | P1.0|-->LED  
//  
// William Goh  
// Texas Instruments Inc.  
// June 2016 (updated) | November 2013 (created)  
// Built with CCSv6.1, IAR, Keil, GCC  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#include "msp.h"  
  
int main(void) {  
 WDT\_A->CTL = WDT\_A\_CTL\_PW | // Stop WDT  
 WDT\_A\_CTL\_HOLD;  
  
 // Configure P1  
 P1->SEL1 &= ~BIT0;  
 P1->SEL0 &= ~BIT0;  
 P1->DIR |= BIT0;  
   
 CS->KEY = CS\_KEY\_VAL; // unlock CS registers  
 CS->CTL0 = 0; // clear register CTL0  
 CS->CTL0=CS\_CTL0\_DCORSEL\_4; //24 MHz  
 CS->CTL1 = CS\_CTL1\_SELA\_2 | CS\_CTL1\_SELS\_3 | CS\_CTL1\_SELM\_3; // select clock sources  
 CS->KEY = 0; // lock the CS registers  
   
   
 P1->OUT |= BIT0;  
   
 TIMER\_A0->CCTL[0] = TIMER\_A\_CCTLN\_CCIE; // TACCR0 interrupt enabled  
   
 TIMER\_A0->CCR[0] = 240; //Set CCR0 to 240 for on cycle  
   
 TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK, continuous mode  
 TIMER\_A\_CTL\_MC\_\_CONTINUOUS;  
  
 SCB->SCR |= SCB\_SCR\_SLEEPONEXIT\_Msk; // Enable sleep on exit from ISR  
  
 // Enable global interrupt  
 \_\_enable\_irq();  
  
 NVIC->ISER[0] = 1 << ((TA0\_0\_IRQn) & 31);  
  
 while (1)  
 {  
 \_\_sleep();  
  
 \_\_no\_operation(); // For debugger  
 }  
}  
  
// Timer A0 interrupt service routine  
  
void TA0\_0\_IRQHandler(void) {  
 TIMER\_A0->CCTL[0] &= ~TIMER\_A\_CCTLN\_CCIFG;  
   
 //Check whether LED is on or off  
 if (P1->OUT & 1) { //LED ON  
 TIMER\_A0->CCR[0] += 720; //off timer  
 P1->OUT &= ~BIT0; //turn led off  
 }  
 else { // LED OFF  
 TIMER\_A0->CCR[0] += 240; //on timer  
 P1->OUT |= BIT0; //turn led on  
 }  
}

**ii.) main.c with ISR for 20-second period clock system.**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// MSP432P401 Demo - Timer0\_A3, Toggle P1.0, CCR0 Cont Mode ISR, DCO SMCLK  
//  
// Description: Toggle P1.0 using software and TA\_0 ISR. Timer0\_A is  
// configured for continuous mode, thus the timer overflows when TAR counts  
// to CCR0. In this example, CCR0 is loaded with 50000.  
// ACLK = n/a, MCLK = SMCLK = TACLK = default DCO = ~3MHz  
//  
// MSP432P401x  
// ---------------  
// /|\| |  
// | | |  
// --|RST |  
// | |  
// | P1.0|-->LED  
//  
// William Goh  
// Texas Instruments Inc.  
// June 2016 (updated) | November 2013 (created)  
// Built with CCSv6.1, IAR, Keil, GCC  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#include "msp.h"  
#define CYCLETIMER 60000 //Adjust this for ISR Timer  
  
int numInterrupts;  
  
int main(void) {  
 WDT\_A->CTL = WDT\_A\_CTL\_PW | // Stop WDT  
 WDT\_A\_CTL\_HOLD;  
  
 // Configure LED  
 P2->SEL1 &= ~BIT1;  
 P2->SEL0 &= ~BIT1;  
 P2->DIR |= BIT1;  
   
 CS->KEY = CS\_KEY\_VAL; // unlock CS registers  
 CS->CTL0 = 0; // clear register CTL0  
 CS->CTL0=CS\_CTL0\_DCORSEL\_0; //1.5 MHz  
 CS->CTL1 = CS\_CTL1\_SELA\_2 | CS\_CTL1\_SELS\_3 | CS\_CTL1\_SELM\_3; // select clock sources  
 CS->KEY = 0; // lock the CS registers  
   
   
 P2->OUT |= BIT1;  
   
 TIMER\_A0->CCTL[0] = TIMER\_A\_CCTLN\_CCIE; // TACCR0 interrupt enabled  
   
 TIMER\_A0->CCR[0] = CYCLETIMER; //Set CCR0 for on cycle  
   
 TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK, continuous mode  
 TIMER\_A\_CTL\_MC\_\_CONTINUOUS;  
  
 SCB->SCR |= SCB\_SCR\_SLEEPONEXIT\_Msk; // Enable sleep on exit from ISR  
  
 // Enable global interrupt  
 \_\_enable\_irq();  
 numInterrupts = 0;  
   
 NVIC->ISER[0] = 1 << ((TA0\_0\_IRQn) & 31);  
  
 while (1)  
 {  
 \_\_sleep();  
  
 \_\_no\_operation(); // For debugger  
 }  
}  
  
// Timer A0 interrupt service routine  
  
void TA0\_0\_IRQHandler(void) {  
 TIMER\_A0->CCTL[0] &= ~TIMER\_A\_CCTLN\_CCIFG;  
   
 numInterrupts += 1;  
 TIMER\_A0->CCR[0] += CYCLETIMER; //increment timer  
   
 if (numInterrupts == 250) {  
 numInterrupts = 0;  
 P2->OUT ^= BIT1; //turn led on/off  
 }  
}

***iii.) main.c with for 2-bit counter generation***

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// MSP432P401 Demo - Timer0\_A3, Toggle P1.0, CCR0 Cont Mode ISR, DCO SMCLK

//

// Description: Toggle P1.0 using software and TA\_0 ISR. Timer0\_A is

// configured for continuous mode, thus the timer overflows when TAR counts

// to CCR0. In this example, CCR0 is loaded with 50000.

// ACLK = n/a, MCLK = SMCLK = TACLK = default DCO = ~3MHz

//

// MSP432P401x

// ---------------

// /|\| |

// | | |

// --|RST |

// | |

// | P1.0|-->LED

//

// William Goh

// Texas Instruments Inc.

// June 2016 (updated) | November 2013 (created)

// Built with CCSv6.1, IAR, Keil, GCC

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "msp.h"

#define CYCLETIMER0 750 //Adjust this for lower bit

#define CYCLETIMER1 1500 //Adjust this for upper bit

int main(void) {

WDT\_A->CTL = WDT\_A\_CTL\_PW | // Stop WDT

WDT\_A\_CTL\_HOLD;

// Configure P5.0 P5.1

P5 -> DIR |= BIT0 | BIT1;

P5 -> OUT = 0;

CS->KEY = CS\_KEY\_VAL; // unlock CS registers

CS->CTL0 = 0; // clear register CTL0

CS->CTL0=CS\_CTL0\_DCORSEL\_0; //1.5 MHz

CS->CTL1 = CS\_CTL1\_SELA\_2 | CS\_CTL1\_SELS\_3 | CS\_CTL1\_SELM\_3; // select clock sources

CS->KEY = 0; // lock the CS registers

TIMER\_A0->CCTL[0] = TIMER\_A\_CCTLN\_CCIE; // TACCR0 interrupt enabled

TIMER\_A0->CCTL[1] = TIMER\_A\_CCTLN\_CCIE; // TACCR1 interrupt enabled

TIMER\_A0->CCR[0] = CYCLETIMER0; //Set CCR0 for on cycle

TIMER\_A0->CCR[1] = CYCLETIMER1; //SET CCR1 for cycle

TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK, continuous mode

TIMER\_A\_CTL\_MC\_\_CONTINUOUS;

SCB->SCR |= SCB\_SCR\_SLEEPONEXIT\_Msk; // Enable sleep on exit from ISR

// Enable global interrupt

\_\_enable\_irq();

NVIC->ISER[0] = 1 << ((TA0\_0\_IRQn) & 31);

NVIC->ISER[0] = 1 << ((TA0\_N\_IRQn) & 31);

while (1)

{

\_\_sleep();

\_\_no\_operation(); // For debugger

}

}

// Timer A0 interrupt service routine

void TA0\_0\_IRQHandler(void) {

TIMER\_A0->CCTL[0] &= ~TIMER\_A\_CCTLN\_CCIFG;

TIMER\_A0->CCR[0] += CYCLETIMER0; //increment timer

if (P5->OUT & BIT0)

P5 -> OUT &= ~BIT0;

else

P5-> OUT |= BIT0;

}

void TA0\_N\_IRQHandler(void) {

if (TIMER\_A0->CCTL[1] & TIMER\_A\_CCTLN\_CCIFG)

{

TIMER\_A0->CCTL[1] &= ~TIMER\_A\_CCTLN\_CCIFG;

TIMER\_A0->CCR[1] += CYCLETIMER1; //increment timer

if (P5->OUT & BIT1)

P5 -> OUT &= ~BIT1;

else

P5-> OUT |= BIT1;

}

}