```c
// PeriodicSysTickInts.c
// Runs on LM3S1968
// Use the SysTick timer to request interrupts at a particular period.
// Daniel Valvano
// June 27, 2011

/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to the Arm Cortex M3",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2011

   Program 5.12, section 5.7

 Copyright 2011 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */

// oscilloscope or LED connected to PD0 for period measurement
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "SysTickInts.h"
#include "PLL.h"
#include "lm3s1968.h"
#include "DAC.h"
#include "Sound.h"
#include "Piano.h"
#include "math.h"

unsigned int guile_notes[] = {51, 51, 50, 0, 50, 51, 50, 51, 50, 51, 0, 50, 53, 0, 53, 51, 50, 46,
                              51, 51, 50, 0, 50, 51, 50, 51, 50, 51, 0, 50, 53, 0, 53, 51, 50, 46,
                              36, 36, 38, 39, 41, 43, 43, 41, 46, 44, 43, 44, 38, 39, 0, 46, 38, 41, 44,
 46, 43, 0, 43, 41, 38,
                              36, 36, 38, 39, 41, 43, 43, 41, 46, 44, 43, 44, 38, 39, 0, 46, 38, 41, 44,
 46, 43, 0, 43, 41, 38};
unsigned int guile_times[] = {1250, 625, 625, 625, 625, 5000, 1250, 625, 1250, 625, 625, 1250, 625, 625,
 625, 1250, 1250, 1250,
                              1250, 625, 625, 625, 625, 5000, 1250, 625, 1250, 625, 625, 1250, 625, 625,
 625, 1250, 1250, 1250,
                              5000, 1250, 1250, 625, 1875, 1875, 625, 1250, 2500, 1250, 625, 1875, 1875,
 1875, 1250, 1250, 1250, 1250, 1875, 1875, 1250, 1250, 1250, 1250, 1250,
                              5000, 1250, 1250, 625, 1875, 1875, 625, 1250, 2500, 1250, 625, 1875, 1875,
 1875, 1250, 1250, 1250, 1250, 1875, 1875, 1250, 1250, 1250, 1250, 1250};

void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts
long StartCritical (void);    // previous I bit, disable interrupts
void EndCritical(long sr);    // restore I bit to previous value
void WaitForInterrupt(void);  // low power mode

int main(void){int i;
  // bus clock at 50 MHz
  //SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN);
  PLL_Init();
  SysTick_Init(50000);     // initialize SysTick timer
  EnableInterrupts();

  DAC_Init();
  Sound_Init();
  for (i = 0; i < sizeof(guile_notes)/sizeof(int); i++) {
    Sound_Play_Timing(guile_notes[i], guile_times[i]);
  }
  while(1){
    Piano_In();
    //WaitForInterrupt();
  }
}
```

```c
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "lm3s1968.h"
#include "Sound.h"

//0x2a - UP, 0x20 - DOWN, 0x26 - LEFT, 0x25 - RIGHT, 0x1e - SELECT
unsigned int note;
void Piano_In (void) {
  note = GPIO_PORTG_DATA_R & 0xF8;
  if (note == 0xF0) { // UP
    Sound_Play(0x2a);
  } else if (note == 0xE8) {  // DOWN
    Sound_Play(0x20);
  } else if (note == 0xD8) {  // LEFT
    Sound_Play(0x26);
  } else if (note == 0xB8) {  // RIGHT
    Sound_Play(0x25);
  } else if (note == 0x78) {  // SELECT
    Sound_Play(0x1e);
  } else {
    Sound_Off();
  }
}

```

```c

// Take input from buttons, send data to Sound.c
void Piano_In(void);

```

```c

#include "Sound.h"
#include <lm3s1968.h>
#include <math.h>
#include "DAC.h"
#include "systick.h"
#include "systickints.h"

const unsigned long SAMPLE_RATE = 128;
unsigned int sinArray[SAMPLE_RATE];
volatile unsigned long index = 0;

void Sound_Init() {
  int sinResult;
  double pi = 4.0 * atan(1.0);
  int i;
  for (i = 0; i < SAMPLE_RATE; i++) {
    sinResult = (int)(7.0*sin(2.0*pi*i/SAMPLE_RATE)+7.49);
    sinArray[i] = sinResult;
  }
}

void Sound_Off(void) {
  SysTickPeriodSet(0);
}

void Sound_Play(unsigned int n) {
  unsigned long note = Sound_Note_To_Frequency(n);

  unsigned long period = 0.75 * 50000000.0 / (SAMPLE_RATE*note);
  SysTickPeriodSet(period);
}

void Sound_Play_Timing(unsigned int note, unsigned long time) {
  unsigned long mult;
  if (note != 0) {
    Sound_Play(note);
  }
  for (mult = 500000; mult > 0; mult--) {
    for (; time > 0; time--) {
      // stupid wait
    }
  }
  index = 0;
}

unsigned long Sound_Note_To_Frequency(unsigned int n) {
  return pow(2.0,(n-50.0)/12.0)*440.0;
}

```

```c

// Global variables
extern const unsigned long SAMPLE_RATE;
extern unsigned int sinArray[];
extern volatile unsigned long index;

// Turns sound off (sets period to 0)
void Sound_Off(void);

// Initialize sound array
void Sound_Init(void);

// Plays a note
// note will be converted to a frequency by the function
// only pass note number (ex. 0x49 for C)
void Sound_Play(unsigned int);

// Plays a note for a specified time
// Time is in milliseconds
void Sound_Play_Timing(unsigned int, unsigned long);

// Convert the note parameter to a frequency
// Should not be needed outside this class
unsigned long Sound_Note_To_Frequency(unsigned int);

```

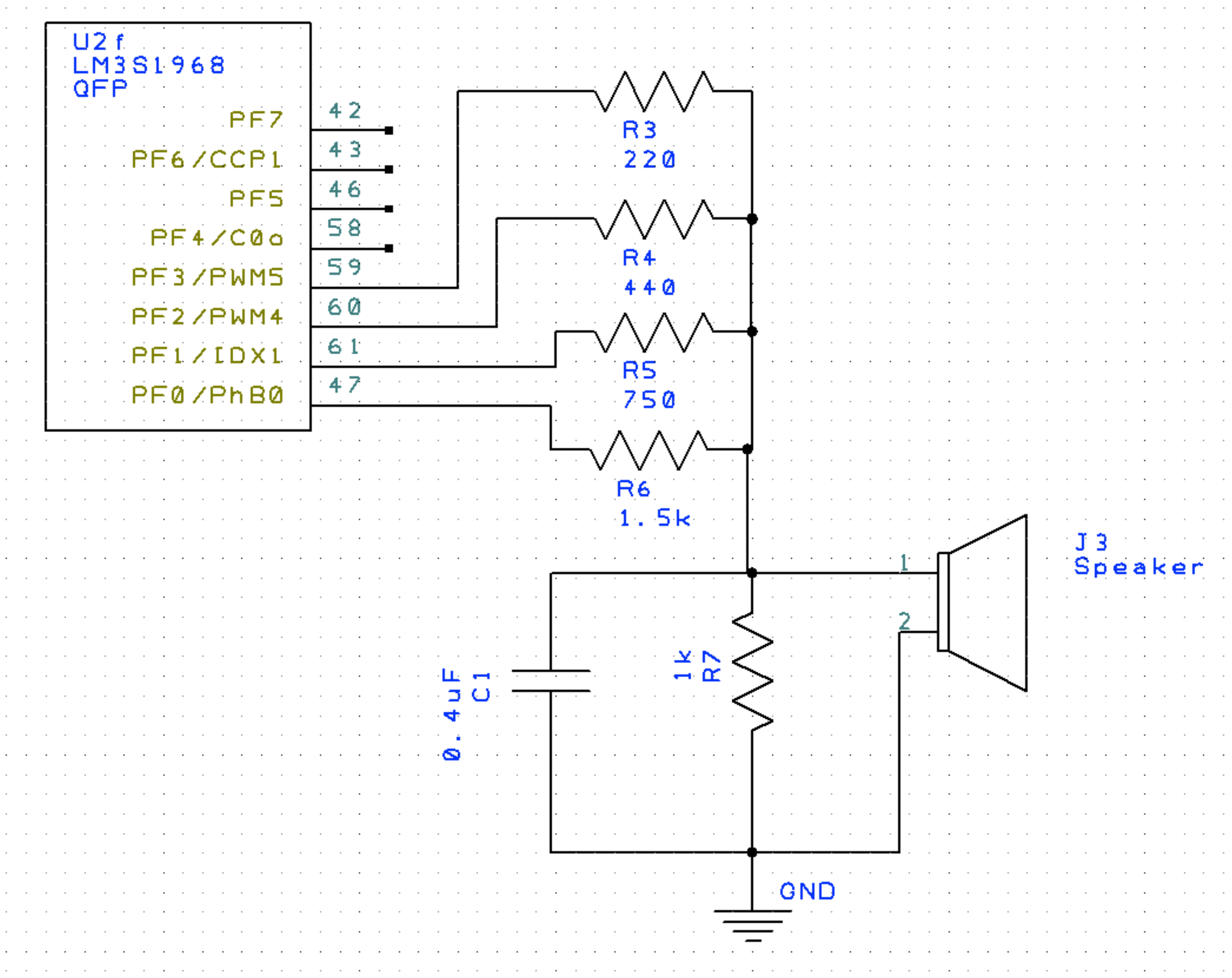
```c
// SysTickInts.c
// Runs on LM3S1968
// Use the SysTick timer to request interrupts at a particular period.
// Daniel Valvano
// June 27, 2011

/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to the Arm Cortex M3",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2011

   Program 5.12, section 5.7

 Copyright 2011 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */

// oscilloscope or LED connected to PD0 for period measurement
#include "hw_types.h"
#include "sysctl.h"
#include "lm3s1968.h"
#include "sound.h"
#include "dac.h"
// #include "lm3s1968.h"

#define NVIC_SYS_PRI3_R         (*((volatile unsigned long *)0xE000ED20))  // Sys. Handlers 12 to 15 
Priority
#define NVIC_ST_CTRL_R          (*((volatile unsigned long *)0xE000E010))
#define NVIC_ST_RELOAD_R        (*((volatile unsigned long *)0xE000E014))
#define NVIC_ST_CURRENT_R       (*((volatile unsigned long *)0xE000E018))
#define NVIC_ST_CTRL_CLK_SRC    0x00000004  // Clock Source
#define NVIC_ST_CTRL_INTEN      0x00000002  // Interrupt enable
#define NVIC_ST_CTRL_ENABLE     0x00000001  // Counter mode
#define GPIO_PORTD_DIR_R        (*((volatile unsigned long *)0x40007400))
#define GPIO_PORTD_DEN_R        (*((volatile unsigned long *)0x4000751C))
#define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE108))

void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts
long StartCritical (void);    // previous I bit, disable interrupts
void EndCritical(long sr);    // restore I bit to previous value
void WaitForInterrupt(void);  // low power mode
#define GPIO_PORTG2             (*((volatile unsigned long *)0x40026010))

// *************SysTick_Init********************
// Initialize Systick periodic interrupts
// Input: interrupt period
//        Units of period are 20ns
//        Maximum is 2^24-1
//        Minimum is determined by length of ISR
// Output: none
void SysTick_Init(unsigned long period){int nop;
  SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOG;
  nop = 0;
  nop = nop + 1;
  GPIO_PORTG_DIR_R &= 0x07;
  GPIO_PORTG_DIR_R |= 0x04;
  GPIO_PORTG_AFSEL_R |= 0xF8;
  GPIO_PORTG_PUR_R |= 0xF8;
  GPIO_PORTG_DEN_R |= 0xFF;
  NVIC_ST_CTRL_R = 0;         // disable SysTick during setup
  NVIC_ST_RELOAD_R = period-1;// reload value
  NVIC_ST_CURRENT_R = 0;      // any write to current clears it
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x40000000; // priority 2
                              // enable SysTick with core clock and interrupts
  NVIC_ST_CTRL_R = NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC+NVIC_ST_CTRL_INTEN;
}

void SysTick_Switch(unsigned int on) {
  if (on == 0) {
    NVIC_ST_CTRL_R = 0;
  } else {
    NVIC_ST_CTRL_R = NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC+NVIC_ST_CTRL_INTEN;
  }
}


void SysTick_Handler(void) {
  GPIO_PORTG2 ^= 0x04;        // toggle PD0
  DAC_Out(sinArray[index]);
  index+=1;
  if (index >= SAMPLE_RATE) {
    index = 0;
  }
}

void SysTick_Wait(unsigned long delay) {
  volatile unsigned long elapsedTime;
  unsigned long startTime = NVIC_ST_CURRENT_R;
  do {
    elapsedTime = (startTime - NVIC_ST_CURRENT_R)&0x00FFFFFF;
  } while (elapsedTime <= delay);
}


```

```c
// SysTickInts.h
// Runs on LM3S1968
// Use the SysTick timer to request interrupts at a particular period.
// Daniel Valvano
// June 27, 2011

/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to the Arm Cortex M3",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2011

   Program 5.12, section 5.7

 Copyright 2011 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */



// **************SysTick_Init*********************
// Initialize Systick periodic interrupts
// Input: interrupt period
//        Units of period are 20ns
//        Maximum is 2^24-1
//        Minimum is determined by length of ISR
// Output: none
void SysTick_Init(unsigned long period);
void SysTick_Switch(unsigned int on);
void SysTick_Wait(unsigned long delay);

```

```c

#include <lm3s1968.h>

void DAC_Init(void) { int nop;
  SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOF;
  nop = 0;
  nop = nop + 1;
  GPIO_PORTF_DIR_R |= 0x0F;
  GPIO_PORTF_DEN_R |= 0x0F;
  GPIO_PORTF_AFSEL_R |= 0x00;
}

void DAC_Out(unsigned long packet) {
  packet &= 0x0F;
  GPIO_PORTF_DATA_R = packet;
}

```

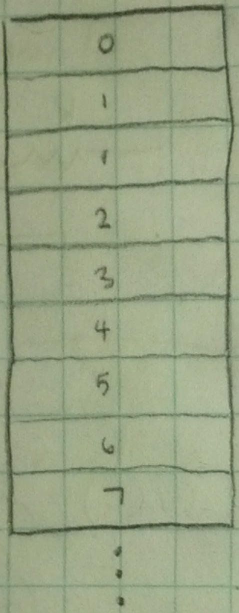
```c

void DAC_Init(void);

void DAC_Out(unsigned long packet);

```

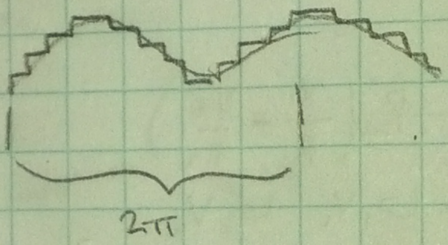
bits	theoretical	measured
0000	0.0	0.0
0001	0.067	0.158
0010	0.135	0.293
0011	0.204	0.455
0100	0.231	0.437
0101	0.299	0.603
0110	0.367	0.747
0111	0.435	0.916
1000	0.462	0.682
1001	0.530	0.857
1010	0.599	1.006
1011	0.666	1.176
1100	0.694	1.163
1101	0.762	1.330
1110	0.830	1.474
1111	0.898	1.636



U2f
LM3S1968
QFP
PF7
PF6/CCP1
PF5
PF4/C0o
PF3/PWM5
PF2/PWM4
PF1/IDX1
PF0/PhB0
42
43
46
58
59
60
61
47
R3
220
R4
440
R5
750
R6
1.5k
1k
R7
0.4uF
C1
1
2
J3
Speaker
GND

Simple Array

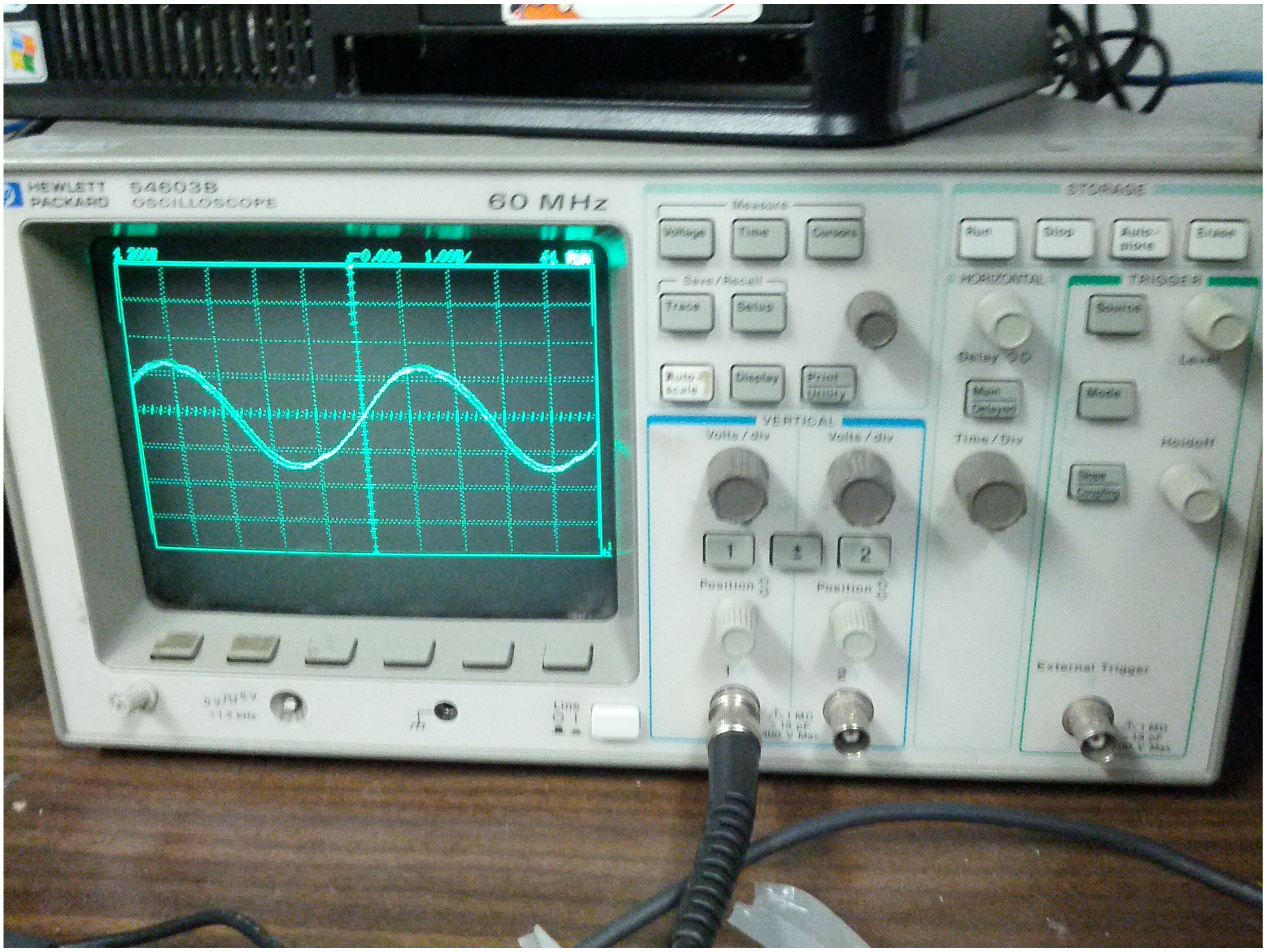




Bit-rate is variable
currently set to 128 Hz

Range of frequencies 0 Hz - 5 MHz

Accuracy: within 10 cents (music)
Precision: < 1% error


HEWLETT PACKARD
54603B OSCILLOSCOPE
60 MHz
Measure
Voltage
Time
Cursors
Save / Recall
Trace
Setup
Auto-scale
Display
Print Utility
STORAGE
Run
Stop
Auto-store
Erase
HORIZONTAL
Delay
Main Delayed
Time / Div
TRIGGER
Source
Level
Mode
Holdoff
Slope Coupling
VERTICAL
Volts / div
Volts / div
1
2
Position
Position
External Trigger
Line