ECE-CT580 Spring 19-20
Homework/Reading Week 2

Please submit the filled in cover sheet as the first page of you HW submission.

Last Name:_Nguyen____________________ First Name:_Tai________________

Students must indicate the status of each problem by:

- **C:** completed,
- **P:** Partially completed,
- **N:** not attempted

Instructor Problem

Problem	Status	Grade/Comments
HW 1 Q Math	C	
HW 2 Ultrasonic Sensor Digging deeper	C	
Optional XC Sketches Demonstrating commands	C	Optional Extra 2 points max
Instructor 1 Fixed point computations	C	
Instructor 2 First cut on Untrasonic Sensor	C	

Final Score:_(10 points)

Look up “Q Math”or Q Arithmetic (Wikipedia is a good first look) Focus on TI processors. Write a maximum 1 page overview of what it is, give simple examples. Do this in your own words and compare to what we did in class. What does the TI processor do to support Q arithmetic?

Answer:

1. What is Q Math?

Q Math stands for fixed-point math, or fixed-point arithmetic (FPA). FPA is a way to represent and approximate fractional numbers in a format that can use leverage the low-cost and simple fixed-point arithmetic units (in hardware) to do additions/multiplication. The alternative to fixed-point unit is floating-point unit, which is substantially more complex. The fixed-point format is often referred to as the xQN format, where x is the number if bits before the decimal point and N is the number of bits after the decimal point. For example, an 8 bit signed decimal number can be represented as 4Q3, meaning, the first bit is the sign, the following 4 bits represent the integer part, and the last 3 bits represent the fractional part (all in binary). A number such as 4.75 can be represented as: 0(positive) 0100(bin for 4) 110 (2^-1 + 2^-2 = 0.5+0.25 = 0.75).

Adding 2 xQN numbers will result in at most (x+1)QN number. Multiplying 2 xQN numbers will result in a (2x)Q(2N) number.

2. What does the TI processor do to support Q arithmetic?

Source: https://processors.wiki.ti.com/images/8/8c/IQMath_fixed_vs_floating.pdf

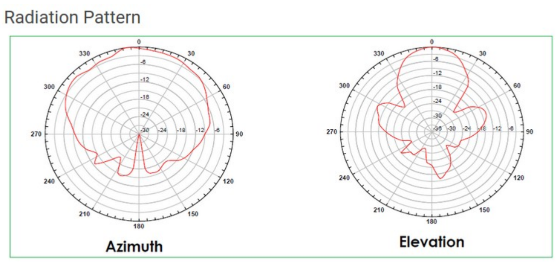
According to the source above, TI processor has no problem doing addition or multiplication with 16 bits number because multiplying 2 16 bits number only results in a 32 bits number, which is the number of bits inside each registers. However, 16 bits number have very limited range and granularity, which is a problem for many systems that can’t handle noise due to bit-truncation. Hence, in order to multiply 2 32 bits number in 8Q24 (which results in a 64 bits number - 16Q48), new instructions will store the lower 32 bits into register P and the upper 32 bits into the ACC register. Then, if we need to add another 8Q24 bit to the result of the multiplication, then the 16Q48 number will be shifted to a 8Q56 number and truncated into a 8Q24 number so that it can be added. These operations are claimed to only take 7 clock cycles (instead of 125 cycles for floating point– IEEE format).

Ultrasonic Sensor:

Model: HC-SR04

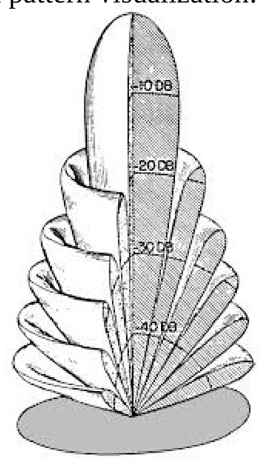
Specs:

- Max Range: 4m
- Min Range: 2cm
- Resolution : 0.3 cm
- Working Temperature: -15°C to 70°C
- Ideal temperature: 20° C
- Temperature Sensitivity: -8.5 cm at 70° C and +7.65 cm at -25° C (reported at https://www.pepperl-fuchs.com/usa/en/25518.htm). However, it's seen worse number (at https://www.instructables.com/id/Temperature-Compensated-Ultrasonic-Range-Finder-Wi/
): +35cm at -20°C.
- Accuracy: at 20° C ~ 1%



http://www.theorycircuit.com/hb100-microwave-motion-sensor-interfacing-arduino/

antenna pattern visualization:



https://www.massa.com/wp-content/uploads/2018/06/Massa-Whitepaper-3-DPM-160621.pdf

/dev/ttyACM0

Send

```
Floating point 6.5625*4.25: 11
Fixed point 6.5625*4.25 (W=8): 1
Floating point 0.5625*30.25: 10
Fixed point 0.5625*30.25 (W=8): 1
Floating point -0.5625*30.25: 10
Fixed point -0.5625*30.25 (W=8): 1
Fixed point 6.5625*4.25 (W=16): 1
Floating point 6.5625*4.25: 11
Fixed point 6.5625*4.25 (W=8): 1
Floating point 0.5625*30.25: 10
Fixed point 0.5625*30.25 (W=8): 1
Floating point -0.5625*30.25: 10
Fixed point -0.5625*30.25 (W=8): 1
Fixed point 6.5625*4.25 (W=16): 1
```

Time to compute floating point is about 10-11x compared to fixed point.

```
volatile float x,y,z;
volatile int8_t a,b,c;

unsigned long t1, t2;

void setup(){
  Serial.begin(9600);

  uint32_t n;
  uint32_t max_it = 10000;

  Serial.print("Floating point 6.5625*4.25: ");
  n = 0;
  t1 = micros();
  while (n < max_it) {
    x = 6.5625;
    y = 4.25;
    z = x*y;
    n++;
  }

  t2 = micros();
  Serial.println((t2-t1)/max_it);

  Serial.print("Fixed point 6.5625*4.25 (W=8):
");
  n = 0;
  t1 = micros();
  while (n < max_it) {
    a = (int8_t)(6.5625*(1 << 3)); // 3Q4
    b = (int8_t)(4.25*(1 << 5));   // 5Q2
    c = (int8_t)(a*b);  // 8Q6 cast into int8
    n++;
  }
  t2 = micros();
  Serial.println((t2-t1)/max_it);

  Serial.print("Floating point 0.5625*30.25: ");
  n = 0;
  t1 = micros();
  while (n < max_it) {
    x = 0.5625;
    y = 30.25;
    z = x*y;
    n++;
  }

  t2 = micros();
  Serial.println((t2-t1)/max_it);
  Serial.print("Fixed point 0.5625*30.25 (W=8):
");
  n = 0;
  t1 = micros();
  while (n < max_it) {
    a = (int8_t)(0.5625*(1 << 3)); // 3Q4
    b = (int8_t)(30.25*(1 << 5));   // 5Q2
    c = (int8_t)(a*b);  // 8Q6 cast into int8
    n++;
  }
  t2 = micros();
  Serial.println((t2-t1)/max_it);
  Serial.print("Floating point -0.5625*30.25:
");
  n = 0;
  t1 = micros();
  while (n < max_it) {
    x = -0.5625;
    y = 30.25;
    z = x*y;
    n++;
  }

  t2 = micros();
  Serial.println((t2-t1)/max_it);

  Serial.print("Fixed point -0.5625*30.25 (W=8):
");
  n = 0;
  t1 = micros();
  while (n < max_it) {
    a = (int8_t)(-0.5625*(1 << 3)); // 3Q4
    b = (int8_t)(30.25*(1 << 5));   // 5Q2
    c = (int8_t)(a*b);  // 8Q6 cast into int8
    n++;
  }
  t2 = micros();
  Serial.println((t2-t1)/max_it);

  Serial.print("Fixed point 6.5625*4.25 (W=16):
");
  n = 0;
  t1 = micros();
  while (n < max_it) {
    a = (int16_t)(6.5625*(1 << 3));
    b = (int16_t)(4.25*(1 << 5));
    c = (int16_t)(a*b);
    n++;
  }
  t2 = micros();
  Serial.println((t2-t1)/max_it);

}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Ultrasonic code using pulseIn()

```
const int trigPin = 4;
const int echoPin = 2;

int duration;
float distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
//  distance = (duration*.0343)/2;
  distance = (duration/2)/29.1;
  Serial.print("Sensor A  ");
  Serial.print(duration);
  Serial.print('\t');
  Serial.println(distance, 4);
  delay(100);
}
```

Ultrasonic code using attachInterrupt()

```
volatile unsigned long LastPulseTime;
volatile unsigned long startTime;
int duration;

#define trigPin 4
#define echoPin 2


void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  attachInterrupt(0, EchoPin_ISR, CHANGE);  // Pin 2 interrupt on
any change
}
void loop(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  Serial.print("Sensor A  ");
  Serial.print(LastPulseTime);
  Serial.print('\t');
  Serial.print((LastPulseTime/2) / 29.1,4);
  Serial.println("cm");

  delay(100);
}
void EchoPin_ISR() {
    if (digitalRead(echoPin)) // Gone HIGH
        startTime = micros();
    else  // Gone LOW
        LastPulseTime = micros() - startTime;
}
```

pulseIn() code results:

There are **9** duration (us) values recorded: 1178, 1177, 1171, 1172, 1154, 1153, 1151, 1150, 1147. The range is 31(us). True distance is 19.96cm. Hence, **error is ~ 1.41%.**
Fanning in direction of measurement sometimes makes device records **infinity**, but otherwise, distance is **not affected**. Fanning in perpendicular to direction of measurement has **little to 0 effect.**

```
           dur distance
Sensor A  1178 20.2405cm
Sensor A  1178 20.2405cm
Sensor A  1178 20.2405cm
Sensor A  1178 20.2405cm
Sensor A  1153 19.7938cm
Sensor A  1153 19.7938cm
Sensor A  1171 20.1031cm
Sensor A  1147 19.6907cm
Sensor A  1178 20.2405cm
Sensor A  1178 20.2405cm
Sensor A  1153 19.7938cm
Sensor A  1177 20.2062cm
Sensor A  1177 20.2062cm
Sensor A  1177 20.2062cm
Sensor A  1150 19.7594cm
Sensor A  1151 19.7594cm
Sensor A  1177 20.2062cm
Sensor A  1153 19.7938cm
Sensor A  1177 20.2062cm
Sensor A  1178 20.2405cm
Sensor A  1171 20.1031cm
Sensor A  1172 20.1375cm
Sensor A  1153 19.7938cm
Sensor A  1153 19.7938cm
Sensor A  1178 20.2405cm
Sensor A  1154 19.8282cm
Sensor A  1178 20.2405cm
Sensor A  1153 19.7938cm
Sensor A  1153 19.7938cm
Sensor A  1178 20.2405cm
Sensor A  1178 20.2405cm
Sensor A  1178 20.2405cm
```

attachInterrupt() code results:

There are **2** duration (us) values recorded: 1152, 1148. The range is 4(us). True distance is 19.96cm. Hence, **error is ~1.13%.**
Fanning in direction of measurement sometimes makes device records **infinity**, but otherwise, distance is **not affected**. Fanning in perpendicular to direction of measurement has **little to 0 effect.**

```
           dur distance
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
Sensor A  1148 19.7251cm
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
Sensor A  1148 19.7251cm
Sensor A  1148 19.7251cm
Sensor A  1148 19.7251cm
Sensor A  1148 19.7251cm
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
Sensor A  1148 19.7251cm
Sensor A  1152 19.7938cm
Sensor A  1148 19.7251cm
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
Sensor A  1148 19.7251cm
Sensor A  1148 19.7251cm
Sensor A  1148 19.7251cm
Sensor A  1148 19.7251cm
Sensor A  1148 19.7251cm
Sensor A  1148 19.7251cm
Sensor A  1148 19.7251cm
Sensor A  1148 19.7251cm
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
Sensor A  1152 19.7938cm
```

pulseIn() code results:

There are **12** duration (us) values recorded: 2212, 2217, 2218, 2219, 2237, 2241, 2243, 2244, 2261, 2267, 2268, 2293. The range is 31(us). True distance is 38.85cm. Hence, **error is ~ 2.17%.** Fanning in direction of measurement sometimes makes device records **infinity**, but otherwise, distance is **not affected**. Fanning in perpendicular to direction of measurement has **little to 0 effect.**

```
          dur distance
Sensor A  2243 38.5223cm
Sensor A  2293 39.3814cm
Sensor A  2237 38.4192cm
Sensor A  2237 38.4192cm
Sensor A  2243 38.5223cm
Sensor A  2243 38.5223cm
Sensor A  2243 38.5223cm
Sensor A  2219 38.1100cm
Sensor A  2244 38.5567cm
Sensor A  2212 38.0069cm
Sensor A  2237 38.4192cm
Sensor A  2218 38.1100cm
Sensor A  2219 38.1100cm
Sensor A  2218 38.1100cm
Sensor A  2218 38.1100cm
Sensor A  2244 38.5567cm
Sensor A  2218 38.1100cm
Sensor A  2237 38.4192cm
Sensor A  2237 38.4192cm
Sensor A  2243 38.5223cm
Sensor A  2219 38.1100cm
Sensor A  2268 38.9691cm
Sensor A  2267 38.9347cm
Sensor A  2241 38.4880cm
Sensor A  2261 38.8316cm
Sensor A  2217 38.0756cm
Sensor A  2267 38.9347cm
Sensor A  2218 38.1100cm
Sensor A  2267 38.9347cm
Sensor A  2218 38.1100cm
Sensor A  2243 38.5223cm
```

attachInterrupt() code results:

There are **5** duration (us) values recorded: 2228, 2232, 2252, 2256, 2280. The range is 4(us). True distance is 38.85cm. Hence, **error is ~1.13%.** Fanning in direction of measurement sometimes makes device records **infinity**, but otherwise, distance is **not affected**. Fanning in perpendicular to direction of measurement has **little to 0 effect.**

```
          dur distance
Sensor A  2232 38.3505cm
Sensor A  2232 38.3505cm
Sensor A  2252 38.6942cm
Sensor A  2252 38.6942cm
Sensor A  2252 38.6942cm
Sensor A  2232 38.3505cm
Sensor A  2252 38.6942cm
Sensor A  2280 39.1753cm
Sensor A  2252 38.6942cm
Sensor A  2256 38.7629cm
Sensor A  2252 38.6942cm
Sensor A  2232 38.3505cm
Sensor A  2256 38.7629cm
Sensor A  2252 38.6942cm
Sensor A  2228 38.2818cm
Sensor A  2256 38.7629cm
Sensor A  2256 38.7629cm
Sensor A  2256 38.7629cm
Sensor A  2232 38.3505cm
Sensor A  2256 38.7629cm
Sensor A  2256 38.7629cm
Sensor A  2252 38.6942cm
Sensor A  2256 38.7629cm
Sensor A  2256 38.7629cm
Sensor A  2280 39.1753cm
Sensor A  2232 38.3505cm
Sensor A  2232 38.3505cm
Sensor A  2232 38.3505cm
Sensor A  2280 39.1753cm
Sensor A  2256 38.7629cm
Sensor A  2232 38.3505cm
```

Ultrasonic setup