```
#include "stm32g4xx.h"
#include <math.h>

#define PI 3.14159265359
#define SAMPLE_RATE 10000 // Sample rate in Hz
#define COS_FREQ 1000 // Cosine frequency in Hz
#define TIMER_CLOCK_FREQ 170000000 // Timer clock frequency in Hz

volatile uint16_t lookup_table[256]; // Lookup table for cos values

void configure_pins() {
  RCC->AHB2ENR |= RCC_AHB2ENR_GPIOBEN; // Enable GPIOB clock
  GPIOB->MODER &= ~GPIO_MODER_MODE0_Msk; // Clear mode for PB0
  GPIOB->MODER |= GPIO_MODER_MODE0_0; // Set PB0 as output
  GPIOB->OSPEEDR |= GPIO_OSPEEDR_OSPEED0_Msk; // Set PB0 as high speed

  // Configure another pin (e.g., PB1) for timing measurement
  GPIOB->MODER &= ~GPIO_MODER_MODE1_Msk;
  GPIOB->MODER |= GPIO_MODER_MODE1_0;
  GPIOB->OSPEEDR |= GPIO_OSPEEDR_OSPEED1_Msk;
}

void configure_timer() {
  RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN; // Enable TIM2 clock
  TIM2->CR1 = 0; // Set timer to default values
  TIM2->PSC = (TIMER_CLOCK_FREQ / SAMPLE_RATE) - 1; // Set prescaler
  TIM2->ARR = SAMPLE_RATE / COS_FREQ - 1; // Set auto-reload value for desired cosine frequency
  TIM2->DIER |= TIM_DIER_UIE; // Enable update interrupt
  NVIC_EnableIRQ(TIM2_IRQn); // Enable TIM2 IRQ
}

void generate_cos_wave() {
  for (int i = 0; i < 256; i++) {
    lookup_table[i] = (uint16_t)(2047 * cos(2 * PI * i / 256) + 2048); // Fill cosine lookup table
```

```
  }
}

void TIM2_IRQHandler() {
  if (TIM2->SR & TIM_SR_UIF) {
    // Toggle GPIO pin PB1 at the start of the ISR
    GPIOB->ODR ^= GPIO_ODR_OD1;

    static uint16_t phase = 0; // Phase accumulator
    phase += 256 * COS_FREQ / SAMPLE_RATE;
    GPIOB->ODR = (lookup_table[phase >> 8] >> 4) & 0x1; // Output cos value to PB0 (assuming 1-bit resolution)

    // Toggle GPIO pin PB1 at the end of the ISR
    GPIOB->ODR ^= GPIO_ODR_OD1;

    TIM2->SR &= ~TIM_SR_UIF; // Clear timer interrupt flag
  }
}

int main() {
  configure_pins();
  configure_timer();
  generate_cos_wave();
  TIM2->CR1 |= TIM_CR1_CEN; // Enable timer
  while (1) {
    // Main loop
  }
}
```