```
int8_t bmi2_accel_gyro_set_config(struct bmi2_dev *bmi2_dev)
{
  /* Variable to define result */
  int8_t rslt;

  /* Initialize interrupts for gyroscope */
  //uint8_t sens_int = BMI2_DRDY_INT;

  struct bmi2_sens_int_config sens_int = { .type = BMI2_ANY_MOTION, .hw_int_pin =
BMI2_INT1 };

  /* List the sensors which are required to enable */
  uint8_t sens_list[3] = {BMI2_ACCEL, BMI2_GYRO, BMI2_ANY_MOTION};

  /* Structure to define the type of the sensor and its configurations */
  struct bmi2_sens_config config[3];

  /* Configure type of feature */
  config[0].type = BMI2_ACCEL;
  config[1].type = BMI2_GYRO;
  config[2].type = BMI2_ANY_MOTION;


  /* Enable the selected sensors */
  rslt = bmi2_sensor_enable(sens_list, 3, bmi2_dev);

  if (rslt == BMI2_OK)
  {
    /* Get default configurations for the type of feature selected */
    rslt = bmi2_get_sensor_config(config, 3, bmi2_dev);

    if (rslt == BMI2_OK)
    {

      config[0].cfg.acc.odr = BMI2_ACC_ODR_800HZ;

      /* Gravity range of the sensor (+/- 2G, 4G, 8G, 16G). */
      config[0].cfg.acc.range = BMI2_ACC_RANGE_2G;
      config[0].cfg.acc.bwp = BMI2_ACC_NORMAL_AVG4;
      config[0].cfg.acc.filter_perf = BMI2_PERF_OPT_MODE;

      /* The user can change the following configuration parameter according to
their requirement */
      /* Output data Rate. By default ODR is set as 200Hz for gyro */
      config[1].cfg.gyr.odr = BMI2_GYR_ODR_800HZ;
      /* Gyroscope Angular Rate Measurement Range.By default the range is 2000dps
*/
      config[1].cfg.gyr.range = BMI2_GYR_RANGE_2000;
      config[1].cfg.gyr.bwp = BMI2_GYR_NORMAL_MODE;
      config[1].cfg.gyr.noise_perf = BMI2_PERF_OPT_MODE;
      config[1].cfg.gyr.filter_perf = BMI2_PERF_OPT_MODE;
      // any motion feature configuration.

      config[2].cfg.any_motion.select_x = BMI2_ENABLE;
      config[2].cfg.any_motion.select_y = BMI2_ENABLE;
      config[2].cfg.any_motion.select_z = BMI2_ENABLE;
      config[2].cfg.any_motion.threshold = 1;
      config[2].cfg.any_motion.out_conf = 6; //Any motion interrupt is set to
int_status_0 register bit 6.
```

```
      config[2].cfg.any_motion.duration = 1;

      /* Set the sensor configurations */
      rslt = bmi2_set_sensor_config(config, 3, bmi2_dev);

      if (rslt == BMI2_OK)
      {
        /* Map feature interrupt to pins */
        rslt = bmi2_map_feat_int(&sens_int, 1, bmi2_dev);

        //rslt = bmi2_map_data_int(sens_int, BMI2_INT2, bmi2_dev);
      }
    }
  }

  return 0;
}

int8_t IMU_init() {


  /* Variable to define result */
  int8_t rslt;
  i2c_bus = 0x68;
  sensor_data[0].type = BMI2_ACCEL;
  sensor_data[1].type = BMI2_GYRO;

  /* To initialize the hal function */

  Wire.begin();
  Wire.setClock(1000000);
  bmi2.intf_ptr = &i2c_bus;
  bmi2.intf = BMI2_I2C_INTF;
  bmi2.read = bmi2xy_hal_i2c_bus_read;
  bmi2.write = bmi2xy_hal_i2c_bus_write;
  bmi2.read_write_len = 16;
  bmi2.delay_us = bmi2xy_hal_delay_usec;

  /* Config file pointer should be assigned to NULL, so that default file address
is assigned in bmi270_init */
  bmi2.config_file_ptr = NULL;
  uint8_t init_status[1];
  int8_t chip_id;

  /* Initialize bmi270 */
  rslt = bmi270_init(&bmi2);
  //BMI270_read_i2c(0x68, 0x00, init_status, 1);
  rslt = bmi2_accel_gyro_set_config(&bmi2);

  if (rslt != BMI2_OK) return 0;



  delay(250);
  // Interrupt PINs configuration

  struct bmi2_int_pin_config data_int_cfg;
  data_int_cfg.pin_type = BMI2_INT1;
  data_int_cfg.int_latch = BMI2_INT_NON_LATCH;
```

```
  data_int_cfg.pin_cfg[0].output_en = BMI2_INT_OUTPUT_ENABLE; // Output enabled
  data_int_cfg.pin_cfg[0].od = BMI2_INT_PUSH_PULL;      // OpenDrain disabled
  data_int_cfg.pin_cfg[0].lvl = BMI2_INT_ACTIVE_HIGH;      // Signal Active High
  data_int_cfg.pin_cfg[0].input_en = BMI2_INT_INPUT_DISABLE;  // Input Disabled
  rslt =  bmi2_set_int_pin_config( &data_int_cfg, &bmi2);

  if (rslt != BMI2_OK) return 0;

  BMI270_read_i2c(0x68, 0x21, init_status, 1);
  if(init_status[0] != 1) return 0;

  return init_status[0];
}
```