```
// Most of the functionality of this library is based on the VL53L1X API
// provided by ST (STSW-IMG007), and some of the explanatory comments are quoted
// or paraphrased from the API source code, API user manual (UM2356), and
// VL53L1X datasheet.

#include <VL53L1X.h>
#include <Wire.h>

// Constructors ////////////////////////////////////////////////////////////////

VL53L1X::VL53L1X()
  : address(AddressDefault)
  , io_timeout(0) // no timeout
  , did_timeout(false)
  , calibrated(false)
  , saved_vhv_init(0)
  , saved_vhv_timeout(0)
  , distance_mode(Unknown)
{
}

// Public Methods //////////////////////////////////////////////////////////////

void VL53L1X::setAddress(uint8_t new_addr)
{
  writeReg(I2C_SLAVE__DEVICE_ADDRESS, new_addr & 0x7F);
  address = new_addr;
}

// Initialize sensor using settings taken mostly from VL53L1_DataInit() and
// VL53L1_StaticInit().
// If io_2v8 (optional) is true or not given, the sensor is configured for 2V8
// mode.
bool VL53L1X::init(bool io_2v8)
{
  // check model ID and module type registers (values specified in datasheet)
  if (readReg16Bit(IDENTIFICATION__MODEL_ID) != 0xEACC) { return false; }

  // VL53L1_software_reset() begin

  writeReg(SOFT_RESET, 0x00);
  delayMicroseconds(100);
  writeReg(SOFT_RESET, 0x01);

  // give it some time to boot; otherwise the sensor NACKs during the readReg()
  // call below and the Arduino 101 doesn't seem to handle that well
  delay(1);

  // VL53L1_poll_for_boot_completion() begin
```

```
  startTimeout();

  // check last_status in case we still get a NACK to try to deal with it correctly
  while ((readReg(FIRMWARE__SYSTEM_STATUS) & 0x01) == 0 || last_status != 0)
  {
    if (checkTimeoutExpired())
    {
      did_timeout = true;
      return false;
    }
  }
  // VL53L1_poll_for_boot_completion() end

  // VL53L1_software_reset() end

  // VL53L1_DataInit() begin

  // sensor uses 1V8 mode for I/O by default; switch to 2V8 mode if necessary
  if (io_2v8)
  {
    writeReg(PAD_I2C_HV__EXTSUP_CONFIG,
      readReg(PAD_I2C_HV__EXTSUP_CONFIG) | 0x01);
  }

  // store oscillator info for later use
  fast_osc_frequency = readReg16Bit(OSC_MEASURED__FAST_OSC__FREQUENCY);
  osc_calibrate_val = readReg16Bit(RESULT__OSC_CALIBRATE_VAL);

  // VL53L1_DataInit() end

  // VL53L1_StaticInit() begin

  // Note that the API does not actually apply the configuration settings below
  // when VL53L1_StaticInit() is called: it keeps a copy of the sensor's
  // register contents in memory and doesn't actually write them until a
  // measurement is started. Writing the configuration here means we don't have
  // to keep it all in memory and avoids a lot of redundant writes later.

  // the API sets the preset mode to LOWPOWER_AUTONOMOUS here:
  // VL53L1_set_preset_mode() begin

  // VL53L1_preset_mode_standard_ranging() begin

  // values labeled "tuning parm default" are from vl53l1_tuning_parm_defaults.h
  // (API uses these in VL53L1_init_tuning_parm_storage_struct())

  // static config
  // API resets PAD_I2C_HV__EXTSUP_CONFIG here, but maybe we don't want to do
  // that? (seems like it would disable 2V8 mode)
  writeReg16Bit(DSS_CONFIG__TARGET_TOTAL_RATE_MCPS, TargetRate); // should already be this value after reset
```

```
  writeReg(GPIO__TIO_HV_STATUS, 0x02);
  writeReg(SIGMA_ESTIMATOR__EFFECTIVE_PULSE_WIDTH_NS, 8); // tuning parm default
  writeReg(SIGMA_ESTIMATOR__EFFECTIVE_AMBIENT_WIDTH_NS, 16); // tuning parm default
  writeReg(ALGO__CROSSTALK_COMPENSATION_VALID_HEIGHT_MM, 0x01);
  writeReg(ALGO__RANGE_IGNORE_VALID_HEIGHT_MM, 0xFF);
  writeReg(ALGO__RANGE_MIN_CLIP, 0); // tuning parm default
  writeReg(ALGO__CONSISTENCY_CHECK__TOLERANCE, 2); // tuning parm default

  // general config
  writeReg16Bit(SYSTEM__THRESH_RATE_HIGH, 0x0000);
  writeReg16Bit(SYSTEM__THRESH_RATE_LOW, 0x0000);
  writeReg(DSS_CONFIG__APERTURE_ATTENUATION, 0x38);

  // timing config
  // most of these settings will be determined later by distance and timing
  // budget configuration
  writeReg16Bit(RANGE_CONFIG__SIGMA_THRESH, 360); // tuning parm default
  writeReg16Bit(RANGE_CONFIG__MIN_COUNT_RATE_RTN_LIMIT_MCPS, 192); // tuning parm default

  // dynamic config

  writeReg(SYSTEM__GROUPED_PARAMETER_HOLD_0, 0x01);
  writeReg(SYSTEM__GROUPED_PARAMETER_HOLD_1, 0x01);
  writeReg(SD_CONFIG__QUANTIFIER, 2); // tuning parm default

  // VL53L1_preset_mode_standard_ranging() end

  // from VL53L1_preset_mode_timed_ranging_*
  // GPH is 0 after reset, but writing GPH0 and GPH1 above seem to set GPH to 1,
  // and things don't seem to work if we don't set GPH back to 0 (which the API
  // does here).
  writeReg(SYSTEM__GROUPED_PARAMETER_HOLD, 0x00);
  writeReg(SYSTEM__SEED_CONFIG, 1); // tuning parm default

  // from VL53L1_config_low_power_auto_mode
  writeReg(SYSTEM__SEQUENCE_CONFIG, 0x8B); // VHV, PHASECAL, DSS1, RANGE
  writeReg16Bit(DSS_CONFIG__MANUAL_EFFECTIVE_SPADS_SELECT, 200 << 8);
  writeReg(DSS_CONFIG__ROI_MODE_CONTROL, 2); // REQUESTED_EFFFECTIVE_SPADS

  // VL53L1_set_preset_mode() end

  // default to long range, 50 ms timing budget
  // note that this is different than what the API defaults to
  setDistanceMode(Long);
  setMeasurementTimingBudget(50000);

  // VL53L1_StaticInit() end

  // the API triggers this change in VL53L1_init_and_start_range() once a
  // measurement is started; assumes MM1 and MM2 are disabled
```

```
  writeReg16Bit(ALGO__PART_TO_PART_RANGE_OFFSET_MM,
    readReg16Bit(MM_CONFIG__OUTER_OFFSET_MM) * 4);

  return true;
}

// Write an 8-bit register
void VL53L1X::writeReg(uint16_t reg, uint8_t value)
{
  Wire.beginTransmission(address);
  Wire.write((reg >> 8) & 0xFF); // reg high byte
  Wire.write( reg       & 0xFF); // reg low byte
  Wire.write(value);
  last_status = Wire.endTransmission();
}

// Write a 16-bit register
void VL53L1X::writeReg16Bit(uint16_t reg, uint16_t value)
{
  Wire.beginTransmission(address);
  Wire.write((reg >> 8) & 0xFF); // reg high byte
  Wire.write( reg       & 0xFF); // reg low byte
  Wire.write((value >> 8) & 0xFF); // value high byte
  Wire.write( value       & 0xFF); // value low byte
  last_status = Wire.endTransmission();
}

// Write a 32-bit register
void VL53L1X::writeReg32Bit(uint16_t reg, uint32_t value)
{
  Wire.beginTransmission(address);
  Wire.write((reg >> 8) & 0xFF); // reg high byte
  Wire.write( reg       & 0xFF); // reg low byte
  Wire.write((value >> 24) & 0xFF); // value highest byte
  Wire.write((value >> 16) & 0xFF);
  Wire.write((value >>  8) & 0xFF);
  Wire.write( value        & 0xFF); // value lowest byte
  last_status = Wire.endTransmission();
}

// Read an 8-bit register
uint8_t VL53L1X::readReg(regAddr reg)
{
  uint8_t value;

  Wire.beginTransmission(address);
  Wire.write((reg >> 8) & 0xFF); // reg high byte
  Wire.write( reg       & 0xFF); // reg low byte
  last_status = Wire.endTransmission();
```

```
  Wire.requestFrom(address, (uint8_t)1);
  value = Wire.read();

  return value;
}

// Read a 16-bit register
uint16_t VL53L1X::readReg16Bit(uint16_t reg)
{
  uint16_t value;

  Wire.beginTransmission(address);
  Wire.write((reg >> 8) & 0xFF); // reg high byte
  Wire.write( reg       & 0xFF); // reg low byte
  last_status = Wire.endTransmission();

  Wire.requestFrom(address, (uint8_t)2);
  value  = (uint16_t)Wire.read() << 8; // value high byte
  value |=           Wire.read();      // value low byte

  return value;
}

// Read a 32-bit register
uint32_t VL53L1X::readReg32Bit(uint16_t reg)
{
  uint32_t value;

  Wire.beginTransmission(address);
  Wire.write((reg >> 8) & 0xFF); // reg high byte
  Wire.write( reg       & 0xFF); // reg low byte
  last_status = Wire.endTransmission();

  Wire.requestFrom(address, (uint8_t)4);
  value  = (uint32_t)Wire.read() << 24; // value highest byte
  value |= (uint32_t)Wire.read() << 16;
  value |= (uint16_t)Wire.read() <<  8;
  value |=           Wire.read();       // value lowest byte

  return value;
}

// set distance mode to Short, Medium, or Long
// based on VL53L1_SetDistanceMode()
bool VL53L1X::setDistanceMode(DistanceMode mode)
{
  // save existing timing budget
  uint32_t budget_us = getMeasurementTimingBudget();

  switch (mode)
```

```
  {
    case Short:
      // from VL53L1_preset_mode_standard_ranging_short_range()

      // timing config
      writeReg(RANGE_CONFIG__VCSEL_PERIOD_A, 0x07);
      writeReg(RANGE_CONFIG__VCSEL_PERIOD_B, 0x05);
      writeReg(RANGE_CONFIG__VALID_PHASE_HIGH, 0x38);

      // dynamic config
      writeReg(SD_CONFIG__WOI_SD0, 0x07);
      writeReg(SD_CONFIG__WOI_SD1, 0x05);
      writeReg(SD_CONFIG__INITIAL_PHASE_SD0, 6); // tuning parm default
      writeReg(SD_CONFIG__INITIAL_PHASE_SD1, 6); // tuning parm default

      break;

    case Medium:
      // from VL53L1_preset_mode_standard_ranging()

      // timing config
      writeReg(RANGE_CONFIG__VCSEL_PERIOD_A, 0x0B);
      writeReg(RANGE_CONFIG__VCSEL_PERIOD_B, 0x09);
      writeReg(RANGE_CONFIG__VALID_PHASE_HIGH, 0x78);

      // dynamic config
      writeReg(SD_CONFIG__WOI_SD0, 0x0B);
      writeReg(SD_CONFIG__WOI_SD1, 0x09);
      writeReg(SD_CONFIG__INITIAL_PHASE_SD0, 10); // tuning parm default
      writeReg(SD_CONFIG__INITIAL_PHASE_SD1, 10); // tuning parm default

      break;

    case Long: // long
      // from VL53L1_preset_mode_standard_ranging_long_range()

      // timing config
      writeReg(RANGE_CONFIG__VCSEL_PERIOD_A, 0x0F);
      writeReg(RANGE_CONFIG__VCSEL_PERIOD_B, 0x0D);
      writeReg(RANGE_CONFIG__VALID_PHASE_HIGH, 0xB8);

      // dynamic config
      writeReg(SD_CONFIG__WOI_SD0, 0x0F);
      writeReg(SD_CONFIG__WOI_SD1, 0x0D);
      writeReg(SD_CONFIG__INITIAL_PHASE_SD0, 14); // tuning parm default
      writeReg(SD_CONFIG__INITIAL_PHASE_SD1, 14); // tuning parm default

      break;

    default:
```

```
      // unrecognized mode - do nothing
      return false;
  }

  // reapply timing budget
  setMeasurementTimingBudget(budget_us);

  // save mode so it can be returned by getDistanceMode()
  distance_mode = mode;

  return true;
}

// Set the measurement timing budget in microseconds, which is the time allowed
// for one measurement. A longer timing budget allows for more accurate
// measurements.
// based on VL53L1_SetMeasurementTimingBudgetMicroSeconds()
bool VL53L1X::setMeasurementTimingBudget(uint32_t budget_us)
{
  // assumes PresetMode is LOWPOWER_AUTONOMOUS

  if (budget_us <= TimingGuard) { return false; }

  uint32_t range_config_timeout_us = budget_us -= TimingGuard;
  if (range_config_timeout_us > 1100000) { return false; } // FDA_MAX_TIMING_BUDGET_US * 2

  range_config_timeout_us /= 2;

  // VL53L1_calc_timeout_register_values() begin

  uint32_t macro_period_us;

  // "Update Macro Period for Range A VCSEL Period"
  macro_period_us = calcMacroPeriod(readReg(RANGE_CONFIG__VCSEL_PERIOD_A));

  // "Update Phase timeout - uses Timing A"
  // Timeout of 1000 is tuning parm default (TIMED_PHASECAL_CONFIG_TIMEOUT_US_DEFAULT)
  // via VL53L1_get_preset_mode_timing_cfg().
  uint32_t phasecal_timeout_mclks = timeoutMicrosecondsToMclks(1000, macro_period_us);
  if (phasecal_timeout_mclks > 0xFF) { phasecal_timeout_mclks = 0xFF; }
  writeReg(PHASECAL_CONFIG__TIMEOUT_MACROP, phasecal_timeout_mclks);

  // "Update MM Timing A timeout"
  // Timeout of 1 is tuning parm default (LOWPOWERAUTO_MM_CONFIG_TIMEOUT_US_DEFAULT)
  // via VL53L1_get_preset_mode_timing_cfg(). With the API, the register
  // actually ends up with a slightly different value because it gets assigned,
  // retrieved, recalculated with a different macro period, and reassigned,
  // but it probably doesn't matter because it seems like the MM ("mode
  // mitigation"?) sequence steps are disabled in low power auto mode anyway.
  writeReg16Bit(MM_CONFIG__TIMEOUT_MACROP_A, encodeTimeout(
```

```
    timeoutMicrosecondsToMclks(1, macro_period_us)));

  // "Update Range Timing A timeout"
  writeReg16Bit(RANGE_CONFIG__TIMEOUT_MACROP_A, encodeTimeout(
    timeoutMicrosecondsToMclks(range_config_timeout_us, macro_period_us)));

  // "Update Macro Period for Range B VCSEL Period"
  macro_period_us = calcMacroPeriod(readReg(RANGE_CONFIG__VCSEL_PERIOD_B));

  // "Update MM Timing B timeout"
  // (See earlier comment about MM Timing A timeout.)
  writeReg16Bit(MM_CONFIG__TIMEOUT_MACROP_B, encodeTimeout(
    timeoutMicrosecondsToMclks(1, macro_period_us)));

  // "Update Range Timing B timeout"
  writeReg16Bit(RANGE_CONFIG__TIMEOUT_MACROP_B, encodeTimeout(
    timeoutMicrosecondsToMclks(range_config_timeout_us, macro_period_us)));

  // VL53L1_calc_timeout_register_values() end

  return true;
}

// Get the measurement timing budget in microseconds
// based on VL53L1_SetMeasurementTimingBudgetMicroSeconds()
uint32_t VL53L1X::getMeasurementTimingBudget()
{
  // assumes PresetMode is LOWPOWER_AUTONOMOUS and these sequence steps are
  // enabled: VHV, PHASECAL, DSS1, RANGE

  // VL53L1_get_timeouts_us() begin

  // "Update Macro Period for Range A VCSEL Period"
  uint32_t macro_period_us = calcMacroPeriod(readReg(RANGE_CONFIG__VCSEL_PERIOD_A));

  // "Get Range Timing A timeout"

  uint32_t range_config_timeout_us = timeoutMclksToMicroseconds(decodeTimeout(
    readReg16Bit(RANGE_CONFIG__TIMEOUT_MACROP_A)), macro_period_us);

  // VL53L1_get_timeouts_us() end

  return  2 * range_config_timeout_us + TimingGuard;
}

// Start continuous ranging measurements, with the given inter-measurement
// period in milliseconds determining how often the sensor takes a measurement.
void VL53L1X::startContinuous(uint32_t period_ms)
{
  // from VL53L1_set_inter_measurement_period_ms()
```

```
  writeReg32Bit(SYSTEM__INTERMEASUREMENT_PERIOD, period_ms * osc_calibrate_val);

  writeReg(SYSTEM__INTERRUPT_CLEAR, 0x01); // sys_interrupt_clear_range
  writeReg(SYSTEM__MODE_START, 0x40); // mode_range__timed
}

// Stop continuous measurements
// based on VL53L1_stop_range()
void VL53L1X::stopContinuous()
{
  writeReg(SYSTEM__MODE_START, 0x80); // mode_range__abort

  // VL53L1_low_power_auto_data_stop_range() begin

  calibrated = false;

  // "restore vhv configs"
  if (saved_vhv_init != 0)
  {
    writeReg(VHV_CONFIG__INIT, saved_vhv_init);
  }
  if (saved_vhv_timeout != 0)
  {
     writeReg(VHV_CONFIG__TIMEOUT_MACROP_LOOP_BOUND, saved_vhv_timeout);
  }

  // "remove phasecal override"
  writeReg(PHASECAL_CONFIG__OVERRIDE, 0x00);

  // VL53L1_low_power_auto_data_stop_range() end
}

// Returns a range reading in millimeters when continuous mode is active
// (readRangeSingleMillimeters() also calls this function after starting a
// single-shot range measurement)
uint16_t VL53L1X::read(bool blocking)
{
  if (blocking)
  {
    startTimeout();
    while (!dataReady())
    {
      if (checkTimeoutExpired())
      {
        did_timeout = true;
        ranging_data.range_status = None;
        ranging_data.range_mm = 0;
        ranging_data.peak_signal_count_rate_MCPS = 0;
        ranging_data.ambient_count_rate_MCPS = 0;
        return ranging_data.range_mm;
```

```
      }
    }
  }

  readResults();

  if (!calibrated)
  {
    setupManualCalibration();
    calibrated = true;
  }

  updateDSS();

  getRangingData();

  writeReg(SYSTEM__INTERRUPT_CLEAR, 0x01); // sys_interrupt_clear_range

  return ranging_data.range_mm;
}

// convert a RangeStatus to a readable string
// Note that on an AVR, these strings are stored in RAM (dynamic memory), which
// makes working with them easier but uses up 200+ bytes of RAM (many AVR-based
// Arduinos only have about 2000 bytes of RAM). You can avoid this memory usage
// if you do not call this function in your sketch.
const char * VL53L1X::rangeStatusToString(RangeStatus status)
{
  switch (status)
  {
    case RangeValid:
      return "range valid";

    case SigmaFail:
      return "sigma fail";

    case SignalFail:
      return "signal fail";

    case RangeValidMinRangeClipped:
      return "range valid, min range clipped";

    case OutOfBoundsFail:
      return "out of bounds fail";

    case HardwareFail:
      return "hardware fail";

    case RangeValidNoWrapCheckFail:
      return "range valid, no wrap check fail";
```

```
    case WrapTargetFail:
      return "wrap target fail";

    case XtalkSignalFail:
      return "xtalk signal fail";

    case SynchronizationInt:
      return "synchronization int";

    case MinRangeFail:
      return "min range fail";

    case None:
      return "no update";

    default:
      return "unknown status";
  }
}

// Did a timeout occur in one of the read functions since the last call to
// timeoutOccurred()?
bool VL53L1X::timeoutOccurred()
{
  bool tmp = did_timeout;
  did_timeout = false;
  return tmp;
}

// Private Methods /////////////////////////////////////////////////////////////

// "Setup ranges after the first one in low power auto mode by turning off
// FW calibration steps and programming static values"
// based on VL53L1_low_power_auto_setup_manual_calibration()
void VL53L1X::setupManualCalibration()
{
  // "save original vhv configs"
  saved_vhv_init = readReg(VHV_CONFIG__INIT);
  saved_vhv_timeout = readReg(VHV_CONFIG__TIMEOUT_MACROP_LOOP_BOUND);

  // "disable VHV init"
  writeReg(VHV_CONFIG__INIT, saved_vhv_init & 0x7F);

  // "set loop bound to tuning param"
  writeReg(VHV_CONFIG__TIMEOUT_MACROP_LOOP_BOUND,
    (saved_vhv_timeout & 0x03) + (3 << 2)); // tuning parm default (LOWPOWERAUTO_VHV_LOOP_BOUND_DEFAULT)

  // "override phasecal"
  writeReg(PHASECAL_CONFIG__OVERRIDE, 0x01);
```

```
  writeReg(CAL_CONFIG__VCSEL_START, readReg(PHASECAL_RESULT__VCSEL_START));
}

// read measurement results into buffer
void VL53L1X::readResults()
{
  Wire.beginTransmission(address);
  Wire.write((RESULT__RANGE_STATUS >> 8) & 0xFF); // reg high byte
  Wire.write( RESULT__RANGE_STATUS       & 0xFF); // reg low byte
  last_status = Wire.endTransmission();

  Wire.requestFrom(address, (uint8_t)17);

  results.range_status = Wire.read();

  Wire.read(); // report_status: not used

  results.stream_count = Wire.read();

  results.dss_actual_effective_spads_sd0  = (uint16_t)Wire.read() << 8; // high byte
  results.dss_actual_effective_spads_sd0 |=           Wire.read();      // low byte

  Wire.read(); // peak_signal_count_rate_mcps_sd0: not used
  Wire.read();

  results.ambient_count_rate_mcps_sd0  = (uint16_t)Wire.read() << 8; // high byte
  results.ambient_count_rate_mcps_sd0 |=           Wire.read();      // low byte

  Wire.read(); // sigma_sd0: not used
  Wire.read();

  Wire.read(); // phase_sd0: not used
  Wire.read();

  results.final_crosstalk_corrected_range_mm_sd0  = (uint16_t)Wire.read() << 8; // high byte
  results.final_crosstalk_corrected_range_mm_sd0 |=           Wire.read();      // low byte

  results.peak_signal_count_rate_crosstalk_corrected_mcps_sd0  = (uint16_t)Wire.read() << 8; // high byte
  results.peak_signal_count_rate_crosstalk_corrected_mcps_sd0 |=           Wire.read();      // low byte
}

// perform Dynamic SPAD Selection calculation/update
// based on VL53L1_low_power_auto_update_DSS()
void VL53L1X::updateDSS()
{
  uint16_t spadCount = results.dss_actual_effective_spads_sd0;

  if (spadCount != 0)
  {
    // "Calc total rate per spad"
```

```
    uint32_t totalRatePerSpad =
      (uint32_t)results.peak_signal_count_rate_crosstalk_corrected_mcps_sd0 +
      results.ambient_count_rate_mcps_sd0;

    // "clip to 16 bits"
    if (totalRatePerSpad > 0xFFFF) { totalRatePerSpad = 0xFFFF; }

    // "shift up to take advantage of 32 bits"
    totalRatePerSpad <<= 16;

    totalRatePerSpad /= spadCount;

    if (totalRatePerSpad != 0)
    {
      // "get the target rate and shift up by 16"
      uint32_t requiredSpads = ((uint32_t)TargetRate << 16) / totalRatePerSpad;

      // "clip to 16 bit"
      if (requiredSpads > 0xFFFF) { requiredSpads = 0xFFFF; }

      // "override DSS config"
      writeReg16Bit(DSS_CONFIG__MANUAL_EFFECTIVE_SPADS_SELECT, requiredSpads);
      // DSS_CONFIG__ROI_MODE_CONTROL should already be set to REQUESTED_EFFFECTIVE_SPADS

      return;
    }
  }

  // If we reached this point, it means something above would have resulted in a
  // divide by zero.
  // "We want to gracefully set a spad target, not just exit with an error"

   // "set target to mid point"
   writeReg16Bit(DSS_CONFIG__MANUAL_EFFECTIVE_SPADS_SELECT, 0x8000);
}

// get range, status, rates from results buffer
// based on VL53L1_GetRangingMeasurementData()
void VL53L1X::getRangingData()
{
  // VL53L1_copy_sys_and_core_results_to_range_results() begin

  uint16_t range = results.final_crosstalk_corrected_range_mm_sd0;

  // "apply correction gain"
  // gain factor of 2011 is tuning parm default (VL53L1_TUNINGPARM_LITE_RANGING_GAIN_FACTOR_DEFAULT)
  // Basically, this appears to scale the result by 2011/2048, or about 98%
  // (with the 1024 added for proper rounding).
  ranging_data.range_mm = ((uint32_t)range * 2011 + 0x0400) / 0x0800;
```

```
  // VL53L1_copy_sys_and_core_results_to_range_results() end

  // set range_status in ranging_data based on value of RESULT__RANGE_STATUS register
  // mostly based on ConvertStatusLite()
  switch(results.range_status)
  {
    case 17: // MULTCLIPFAIL
    case 2: // VCSELWATCHDOGTESTFAILURE
    case 1: // VCSELCONTINUITYTESTFAILURE
    case 3: // NOVHVVALUEFOUND
      // from SetSimpleData()
      ranging_data.range_status = HardwareFail;
      break;

    case 13: // USERROICLIP
     // from SetSimpleData()
      ranging_data.range_status = MinRangeFail;
      break;

    case 18: // GPHSTREAMCOUNT0READY
      ranging_data.range_status = SynchronizationInt;
      break;

    case 5: // RANGEPHASECHECK
      ranging_data.range_status =  OutOfBoundsFail;
      break;

    case 4: // MSRCNOTARGET
      ranging_data.range_status = SignalFail;
      break;

    case 6: // SIGMATHRESHOLDCHECK
      ranging_data.range_status = SignalFail;
      break;

    case 7: // PHASECONSISTENCY
      ranging_data.range_status = WrapTargetFail;
      break;

    case 12: // RANGEIGNORETHRESHOLD
      ranging_data.range_status = XtalkSignalFail;
      break;

    case 8: // MINCLIP
      ranging_data.range_status = RangeValidMinRangeClipped;
      break;

    case 9: // RANGECOMPLETE
      // from VL53L1_copy_sys_and_core_results_to_range_results()
```

```
      if (results.stream_count == 0)
      {
        ranging_data.range_status = RangeValidNoWrapCheckFail;
      }
      else
      {
        ranging_data.range_status = RangeValid;
      }
      break;

    default:
      ranging_data.range_status = None;
  }

  // from SetSimpleData()
  ranging_data.peak_signal_count_rate_MCPS =
    countRateFixedToFloat(results.peak_signal_count_rate_crosstalk_corrected_mcps_sd0);
  ranging_data.ambient_count_rate_MCPS =
    countRateFixedToFloat(results.ambient_count_rate_mcps_sd0);
}

// Decode sequence step timeout in MCLKs from register value
// based on VL53L1_decode_timeout()
uint32_t VL53L1X::decodeTimeout(uint16_t reg_val)
{
  return ((uint32_t)(reg_val & 0xFF) << (reg_val >> 8)) + 1;
}

// Encode sequence step timeout register value from timeout in MCLKs
// based on VL53L1_encode_timeout()
uint16_t VL53L1X::encodeTimeout(uint32_t timeout_mclks)
{
  // encoded format: "(LSByte * 2^MSByte) + 1"

  uint32_t ls_byte = 0;
  uint16_t ms_byte = 0;

  if (timeout_mclks > 0)
  {
    ls_byte = timeout_mclks - 1;

    while ((ls_byte & 0xFFFFFF00) > 0)
    {
      ls_byte >>= 1;
      ms_byte++;
    }

    return (ms_byte << 8) | (ls_byte & 0xFF);
  }
  else { return 0; }
```

```
}

// Convert sequence step timeout from macro periods to microseconds with given
// macro period in microseconds (12.12 format)
// based on VL53L1_calc_timeout_us()
uint32_t VL53L1X::timeoutMclksToMicroseconds(uint32_t timeout_mclks, uint32_t macro_period_us)
{
  return ((uint64_t)timeout_mclks * macro_period_us + 0x800) >> 12;
}

// Convert sequence step timeout from microseconds to macro periods with given
// macro period in microseconds (12.12 format)
// based on VL53L1_calc_timeout_mclks()
uint32_t VL53L1X::timeoutMicrosecondsToMclks(uint32_t timeout_us, uint32_t macro_period_us)
{
  return (((uint32_t)timeout_us << 12) + (macro_period_us >> 1)) / macro_period_us;
}

// Calculate macro period in microseconds (12.12 format) with given VCSEL period
// assumes fast_osc_frequency has been read and stored
// based on VL53L1_calc_macro_period_us()
uint32_t VL53L1X::calcMacroPeriod(uint8_t vcsel_period)
{
  // from VL53L1_calc_pll_period_us()
  // fast osc frequency in 4.12 format; PLL period in 0.24 format
  uint32_t pll_period_us = ((uint32_t)0x01 << 30) / fast_osc_frequency;

  // from VL53L1_decode_vcsel_period()
  uint8_t vcsel_period_pclks = (vcsel_period + 1) << 1;

  // VL53L1_MACRO_PERIOD_VCSEL_PERIODS = 2304
  uint32_t macro_period_us = (uint32_t)2304 * pll_period_us;
  macro_period_us >>= 6;
  macro_period_us *= vcsel_period_pclks;
  macro_period_us >>= 6;

  return macro_period_us;
}
```

```
#pragma once

#include <Arduino.h>

class VL53L1X
{
  public:

    // register addresses from API vl53l1x_register_map.h
    enum regAddr : uint16_t
    {
      SOFT_RESET                                          = 0x0000,
      I2C_SLAVE__DEVICE_ADDRESS                           = 0x0001,
      ANA_CONFIG__VHV_REF_SEL_VDDPIX                      = 0x0002,
      ANA_CONFIG__VHV_REF_SEL_VQUENCH                     = 0x0003,
      ANA_CONFIG__REG_AVDD1V2_SEL                         = 0x0004,
      ANA_CONFIG__FAST_OSC__TRIM                          = 0x0005,
      OSC_MEASURED__FAST_OSC__FREQUENCY                   = 0x0006,
      OSC_MEASURED__FAST_OSC__FREQUENCY_HI                = 0x0006,
      OSC_MEASURED__FAST_OSC__FREQUENCY_LO                = 0x0007,
      VHV_CONFIG__TIMEOUT_MACROP_LOOP_BOUND               = 0x0008,
      VHV_CONFIG__COUNT_THRESH                            = 0x0009,
      VHV_CONFIG__OFFSET                                  = 0x000A,
      VHV_CONFIG__INIT                                    = 0x000B,
      GLOBAL_CONFIG__SPAD_ENABLES_REF_0                   = 0x000D,
      GLOBAL_CONFIG__SPAD_ENABLES_REF_1                   = 0x000E,
      GLOBAL_CONFIG__SPAD_ENABLES_REF_2                   = 0x000F,
      GLOBAL_CONFIG__SPAD_ENABLES_REF_3                   = 0x0010,
      GLOBAL_CONFIG__SPAD_ENABLES_REF_4                   = 0x0011,
      GLOBAL_CONFIG__SPAD_ENABLES_REF_5                   = 0x0012,
      GLOBAL_CONFIG__REF_EN_START_SELECT                  = 0x0013,
      REF_SPAD_MAN__NUM_REQUESTED_REF_SPADS               = 0x0014,
      REF_SPAD_MAN__REF_LOCATION                          = 0x0015,
      ALGO__CROSSTALK_COMPENSATION_PLANE_OFFSET_KCPS      = 0x0016,
      ALGO__CROSSTALK_COMPENSATION_PLANE_OFFSET_KCPS_HI   = 0x0016,
      ALGO__CROSSTALK_COMPENSATION_PLANE_OFFSET_KCPS_LO   = 0x0017,
      ALGO__CROSSTALK_COMPENSATION_X_PLANE_GRADIENT_KCPS  = 0x0018,
      ALGO__CROSSTALK_COMPENSATION_X_PLANE_GRADIENT_KCPS_HI = 0x0018,
      ALGO__CROSSTALK_COMPENSATION_X_PLANE_GRADIENT_KCPS_LO = 0x0019,
      ALGO__CROSSTALK_COMPENSATION_Y_PLANE_GRADIENT_KCPS  = 0x001A,
      ALGO__CROSSTALK_COMPENSATION_Y_PLANE_GRADIENT_KCPS_HI = 0x001A,
      ALGO__CROSSTALK_COMPENSATION_Y_PLANE_GRADIENT_KCPS_LO = 0x001B,
      REF_SPAD_CHAR__TOTAL_RATE_TARGET_MCPS               = 0x001C,
      REF_SPAD_CHAR__TOTAL_RATE_TARGET_MCPS_HI            = 0x001C,
      REF_SPAD_CHAR__TOTAL_RATE_TARGET_MCPS_LO            = 0x001D,
      ALGO__PART_TO_PART_RANGE_OFFSET_MM                  = 0x001E,
      ALGO__PART_TO_PART_RANGE_OFFSET_MM_HI               = 0x001E,
      ALGO__PART_TO_PART_RANGE_OFFSET_MM_LO               = 0x001F,
      MM_CONFIG__INNER_OFFSET_MM                          = 0x0020,
      MM_CONFIG__INNER_OFFSET_MM_HI                       = 0x0020,
```

```
    MM_CONFIG__INNER_OFFSET_MM_LO                                = 0x0021,
    MM_CONFIG__OUTER_OFFSET_MM                                   = 0x0022,
    MM_CONFIG__OUTER_OFFSET_MM_HI                                = 0x0022,
    MM_CONFIG__OUTER_OFFSET_MM_LO                                = 0x0023,
    DSS_CONFIG__TARGET_TOTAL_RATE_MCPS                           = 0x0024,
    DSS_CONFIG__TARGET_TOTAL_RATE_MCPS_HI                        = 0x0024,
    DSS_CONFIG__TARGET_TOTAL_RATE_MCPS_LO                        = 0x0025,
    DEBUG__CTRL                                                  = 0x0026,
    TEST_MODE__CTRL                                              = 0x0027,
    CLK_GATING__CTRL                                             = 0x0028,
    NVM_BIST__CTRL                                               = 0x0029,
    NVM_BIST__NUM_NVM_WORDS                                      = 0x002A,
    NVM_BIST__START_ADDRESS                                      = 0x002B,
    HOST_IF__STATUS                                              = 0x002C,
    PAD_I2C_HV__CONFIG                                           = 0x002D,
    PAD_I2C_HV__EXTSUP_CONFIG                                    = 0x002E,
    GPIO_HV_PAD__CTRL                                            = 0x002F,
    GPIO_HV_MUX__CTRL                                            = 0x0030,
    GPIO__TIO_HV_STATUS                                          = 0x0031,
    GPIO__FIO_HV_STATUS                                          = 0x0032,
    ANA_CONFIG__SPAD_SEL_PSWIDTH                                 = 0x0033,
    ANA_CONFIG__VCSEL_PULSE_WIDTH_OFFSET                         = 0x0034,
    ANA_CONFIG__FAST_OSC__CONFIG_CTRL                            = 0x0035,
    SIGMA_ESTIMATOR__EFFECTIVE_PULSE_WIDTH_NS                    = 0x0036,
    SIGMA_ESTIMATOR__EFFECTIVE_AMBIENT_WIDTH_NS                  = 0x0037,
    SIGMA_ESTIMATOR__SIGMA_REF_MM                                = 0x0038,
    ALGO__CROSSTALK_COMPENSATION_VALID_HEIGHT_MM                 = 0x0039,
    SPARE_HOST_CONFIG__STATIC_CONFIG_SPARE_0                     = 0x003A,
    SPARE_HOST_CONFIG__STATIC_CONFIG_SPARE_1                     = 0x003B,
    ALGO__RANGE_IGNORE_THRESHOLD_MCPS                            = 0x003C,
    ALGO__RANGE_IGNORE_THRESHOLD_MCPS_HI                         = 0x003C,
    ALGO__RANGE_IGNORE_THRESHOLD_MCPS_LO                         = 0x003D,
    ALGO__RANGE_IGNORE_VALID_HEIGHT_MM                           = 0x003E,
    ALGO__RANGE_MIN_CLIP                                         = 0x003F,
    ALGO__CONSISTENCY_CHECK__TOLERANCE                           = 0x0040,
    SPARE_HOST_CONFIG__STATIC_CONFIG_SPARE_2                     = 0x0041,
    SD_CONFIG__RESET_STAGES_MSB                                  = 0x0042,
    SD_CONFIG__RESET_STAGES_LSB                                  = 0x0043,
    GPH_CONFIG__STREAM_COUNT_UPDATE_VALUE                        = 0x0044,
    GLOBAL_CONFIG__STREAM_DIVIDER                                = 0x0045,
    SYSTEM__INTERRUPT_CONFIG_GPIO                                = 0x0046,
    CAL_CONFIG__VCSEL_START                                      = 0x0047,
    CAL_CONFIG__REPEAT_RATE                                      = 0x0048,
    CAL_CONFIG__REPEAT_RATE_HI                                   = 0x0048,
    CAL_CONFIG__REPEAT_RATE_LO                                   = 0x0049,
    GLOBAL_CONFIG__VCSEL_WIDTH                                   = 0x004A,
    PHASECAL_CONFIG__TIMEOUT_MACROP                              = 0x004B,
    PHASECAL_CONFIG__TARGET                                      = 0x004C,
    PHASECAL_CONFIG__OVERRIDE                                    = 0x004D,
    DSS_CONFIG__ROI_MODE_CONTROL                                 = 0x004F,
```

```
      SYSTEM__THRESH_RATE_HIGH                                        = 0x0050,
      SYSTEM__THRESH_RATE_HIGH_HI                                     = 0x0050,
      SYSTEM__THRESH_RATE_HIGH_LO                                     = 0x0051,
      SYSTEM__THRESH_RATE_LOW                                         = 0x0052,
      SYSTEM__THRESH_RATE_LOW_HI                                      = 0x0052,
      SYSTEM__THRESH_RATE_LOW_LO                                      = 0x0053,
      DSS_CONFIG__MANUAL_EFFECTIVE_SPADS_SELECT                       = 0x0054,
      DSS_CONFIG__MANUAL_EFFECTIVE_SPADS_SELECT_HI                    = 0x0054,
      DSS_CONFIG__MANUAL_EFFECTIVE_SPADS_SELECT_LO                    = 0x0055,
      DSS_CONFIG__MANUAL_BLOCK_SELECT                                 = 0x0056,
      DSS_CONFIG__APERTURE_ATTENUATION                                = 0x0057,
      DSS_CONFIG__MAX_SPADS_LIMIT                                     = 0x0058,
      DSS_CONFIG__MIN_SPADS_LIMIT                                     = 0x0059,
      MM_CONFIG__TIMEOUT_MACROP_A                                     = 0x005A, // added by Pololu for 16-b
it accesses
      MM_CONFIG__TIMEOUT_MACROP_A_HI                                  = 0x005A,
      MM_CONFIG__TIMEOUT_MACROP_A_LO                                  = 0x005B,
      MM_CONFIG__TIMEOUT_MACROP_B                                     = 0x005C, // added by Pololu for 16-b
it accesses
      MM_CONFIG__TIMEOUT_MACROP_B_HI                                  = 0x005C,
      MM_CONFIG__TIMEOUT_MACROP_B_LO                                  = 0x005D,
      RANGE_CONFIG__TIMEOUT_MACROP_A                                  = 0x005E, // added by Pololu for 16-b
it accesses
      RANGE_CONFIG__TIMEOUT_MACROP_A_HI                               = 0x005E,
      RANGE_CONFIG__TIMEOUT_MACROP_A_LO                               = 0x005F,
      RANGE_CONFIG__VCSEL_PERIOD_A                                    = 0x0060,
      RANGE_CONFIG__TIMEOUT_MACROP_B                                  = 0x0061, // added by Pololu for 16-b
it accesses
      RANGE_CONFIG__TIMEOUT_MACROP_B_HI                               = 0x0061,
      RANGE_CONFIG__TIMEOUT_MACROP_B_LO                               = 0x0062,
      RANGE_CONFIG__VCSEL_PERIOD_B                                    = 0x0063,
      RANGE_CONFIG__SIGMA_THRESH                                      = 0x0064,
      RANGE_CONFIG__SIGMA_THRESH_HI                                   = 0x0064,
      RANGE_CONFIG__SIGMA_THRESH_LO                                   = 0x0065,
      RANGE_CONFIG__MIN_COUNT_RATE_RTN_LIMIT_MCPS                     = 0x0066,
      RANGE_CONFIG__MIN_COUNT_RATE_RTN_LIMIT_MCPS_HI                  = 0x0066,
      RANGE_CONFIG__MIN_COUNT_RATE_RTN_LIMIT_MCPS_LO                  = 0x0067,
      RANGE_CONFIG__VALID_PHASE_LOW                                   = 0x0068,
      RANGE_CONFIG__VALID_PHASE_HIGH                                  = 0x0069,
      SYSTEM__INTERMEASUREMENT_PERIOD                                 = 0x006C,
      SYSTEM__INTERMEASUREMENT_PERIOD_3                               = 0x006C,
      SYSTEM__INTERMEASUREMENT_PERIOD_2                               = 0x006D,
      SYSTEM__INTERMEASUREMENT_PERIOD_1                               = 0x006E,
      SYSTEM__INTERMEASUREMENT_PERIOD_0                               = 0x006F,
      SYSTEM__FRACTIONAL_ENABLE                                       = 0x0070,
      SYSTEM__GROUPED_PARAMETER_HOLD_0                                = 0x0071,
      SYSTEM__THRESH_HIGH                                             = 0x0072,
      SYSTEM__THRESH_HIGH_HI                                          = 0x0072,
      SYSTEM__THRESH_HIGH_LO                                          = 0x0073,
      SYSTEM__THRESH_LOW                                              = 0x0074,
```

```
    SYSTEM__THRESH_LOW_HI                                                = 0x0074,
    SYSTEM__THRESH_LOW_LO                                                = 0x0075,
    SYSTEM__ENABLE_XTALK_PER_QUADRANT                                    = 0x0076,
    SYSTEM__SEED_CONFIG                                                  = 0x0077,
    SD_CONFIG__WOI_SD0                                                   = 0x0078,
    SD_CONFIG__WOI_SD1                                                   = 0x0079,
    SD_CONFIG__INITIAL_PHASE_SD0                                         = 0x007A,
    SD_CONFIG__INITIAL_PHASE_SD1                                         = 0x007B,
    SYSTEM__GROUPED_PARAMETER_HOLD_1                                     = 0x007C,
    SD_CONFIG__FIRST_ORDER_SELECT                                        = 0x007D,
    SD_CONFIG__QUANTIFIER                                                = 0x007E,
    ROI_CONFIG__USER_ROI_CENTRE_SPAD                                     = 0x007F,
    ROI_CONFIG__USER_ROI_REQUESTED_GLOBAL_XY_SIZE                        = 0x0080,
    SYSTEM__SEQUENCE_CONFIG                                              = 0x0081,
    SYSTEM__GROUPED_PARAMETER_HOLD                                       = 0x0082,
    POWER_MANAGEMENT__GO1_POWER_FORCE                                    = 0x0083,
    SYSTEM__STREAM_COUNT_CTRL                                            = 0x0084,
    FIRMWARE__ENABLE                                                     = 0x0085,
    SYSTEM__INTERRUPT_CLEAR                                              = 0x0086,
    SYSTEM__MODE_START                                                   = 0x0087,
    RESULT__INTERRUPT_STATUS                                             = 0x0088,
    RESULT__RANGE_STATUS                                                 = 0x0089,
    RESULT__REPORT_STATUS                                                = 0x008A,
    RESULT__STREAM_COUNT                                                 = 0x008B,
    RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD0                               = 0x008C,
    RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD0_HI                            = 0x008C,
    RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD0_LO                            = 0x008D,
    RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD0                              = 0x008E,
    RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD0_HI                           = 0x008E,
    RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD0_LO                           = 0x008F,
    RESULT__AMBIENT_COUNT_RATE_MCPS_SD0                                  = 0x0090,
    RESULT__AMBIENT_COUNT_RATE_MCPS_SD0_HI                               = 0x0090,
    RESULT__AMBIENT_COUNT_RATE_MCPS_SD0_LO                               = 0x0091,
    RESULT__SIGMA_SD0                                                    = 0x0092,
    RESULT__SIGMA_SD0_HI                                                 = 0x0092,
    RESULT__SIGMA_SD0_LO                                                 = 0x0093,
    RESULT__PHASE_SD0                                                    = 0x0094,
    RESULT__PHASE_SD0_HI                                                 = 0x0094,
    RESULT__PHASE_SD0_LO                                                 = 0x0095,
    RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD0                       = 0x0096,
    RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD0_HI                    = 0x0096,
    RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD0_LO                    = 0x0097,
    RESULT__PEAK_SIGNAL_COUNT_RATE_CROSSTALK_CORRECTED_MCPS_SD0          = 0x0098,
    RESULT__PEAK_SIGNAL_COUNT_RATE_CROSSTALK_CORRECTED_MCPS_SD0_HI       = 0x0098,
    RESULT__PEAK_SIGNAL_COUNT_RATE_CROSSTALK_CORRECTED_MCPS_SD0_LO       = 0x0099,
    RESULT__MM_INNER_ACTUAL_EFFECTIVE_SPADS_SD0                          = 0x009A,
    RESULT__MM_INNER_ACTUAL_EFFECTIVE_SPADS_SD0_HI                       = 0x009A,
    RESULT__MM_INNER_ACTUAL_EFFECTIVE_SPADS_SD0_LO                       = 0x009B,
    RESULT__MM_OUTER_ACTUAL_EFFECTIVE_SPADS_SD0                          = 0x009C,
    RESULT__MM_OUTER_ACTUAL_EFFECTIVE_SPADS_SD0_HI                       = 0x009C,
```

```
      RESULT__MM_OUTER_ACTUAL_EFFECTIVE_SPADS_SD0_LO                  = 0x009D,
      RESULT__AVG_SIGNAL_COUNT_RATE_MCPS_SD0                          = 0x009E,
      RESULT__AVG_SIGNAL_COUNT_RATE_MCPS_SD0_HI                       = 0x009E,
      RESULT__AVG_SIGNAL_COUNT_RATE_MCPS_SD0_LO                       = 0x009F,
      RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD1                          = 0x00A0,
      RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD1_HI                       = 0x00A0,
      RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD1_LO                       = 0x00A1,
      RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD1                         = 0x00A2,
      RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD1_HI                      = 0x00A2,
      RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD1_LO                      = 0x00A3,
      RESULT__AMBIENT_COUNT_RATE_MCPS_SD1                             = 0x00A4,
      RESULT__AMBIENT_COUNT_RATE_MCPS_SD1_HI                          = 0x00A4,
      RESULT__AMBIENT_COUNT_RATE_MCPS_SD1_LO                          = 0x00A5,
      RESULT__SIGMA_SD1                                               = 0x00A6,
      RESULT__SIGMA_SD1_HI                                            = 0x00A6,
      RESULT__SIGMA_SD1_LO                                            = 0x00A7,
      RESULT__PHASE_SD1                                               = 0x00A8,
      RESULT__PHASE_SD1_HI                                            = 0x00A8,
      RESULT__PHASE_SD1_LO                                            = 0x00A9,
      RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD1                  = 0x00AA,
      RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD1_HI               = 0x00AA,
      RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD1_LO               = 0x00AB,
      RESULT__SPARE_0_SD1                                             = 0x00AC,
      RESULT__SPARE_0_SD1_HI                                          = 0x00AC,
      RESULT__SPARE_0_SD1_LO                                          = 0x00AD,
      RESULT__SPARE_1_SD1                                             = 0x00AE,
      RESULT__SPARE_1_SD1_HI                                          = 0x00AE,
      RESULT__SPARE_1_SD1_LO                                          = 0x00AF,
      RESULT__SPARE_2_SD1                                             = 0x00B0,
      RESULT__SPARE_2_SD1_HI                                          = 0x00B0,
      RESULT__SPARE_2_SD1_LO                                          = 0x00B1,
      RESULT__SPARE_3_SD1                                             = 0x00B2,
      RESULT__THRESH_INFO                                             = 0x00B3,
      RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0                          = 0x00B4,
      RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_3                        = 0x00B4,
      RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_2                        = 0x00B5,
      RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_1                        = 0x00B6,
      RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_0                        = 0x00B7,
      RESULT_CORE__RANGING_TOTAL_EVENTS_SD0                           = 0x00B8,
      RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_3                         = 0x00B8,
      RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_2                         = 0x00B9,
      RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_1                         = 0x00BA,
      RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_0                         = 0x00BB,
      RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0                            = 0x00BC,
      RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_3                          = 0x00BC,
      RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_2                          = 0x00BD,
      RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_1                          = 0x00BE,
      RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_0                          = 0x00BF,
      RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0                          = 0x00C0,
      RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_3                        = 0x00C0,
```

```
      RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_2                    = 0x00C1,
      RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_1                    = 0x00C2,
      RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_0                    = 0x00C3,
      RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1                      = 0x00C4,
      RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_3                    = 0x00C4,
      RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_2                    = 0x00C5,
      RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_1                    = 0x00C6,
      RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_0                    = 0x00C7,
      RESULT_CORE__RANGING_TOTAL_EVENTS_SD1                       = 0x00C8,
      RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_3                     = 0x00C8,
      RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_2                     = 0x00C9,
      RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_1                     = 0x00CA,
      RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_0                     = 0x00CB,
      RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1                        = 0x00CC,
      RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_3                      = 0x00CC,
      RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_2                      = 0x00CD,
      RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_1                      = 0x00CE,
      RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_0                      = 0x00CF,
      RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1                      = 0x00D0,
      RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_3                    = 0x00D0,
      RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_2                    = 0x00D1,
      RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_1                    = 0x00D2,
      RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_0                    = 0x00D3,
      RESULT_CORE__SPARE_0                                        = 0x00D4,
      PHASECAL_RESULT__REFERENCE_PHASE                            = 0x00D6,
      PHASECAL_RESULT__REFERENCE_PHASE_HI                         = 0x00D6,
      PHASECAL_RESULT__REFERENCE_PHASE_LO                         = 0x00D7,
      PHASECAL_RESULT__VCSEL_START                                = 0x00D8,
      REF_SPAD_CHAR_RESULT__NUM_ACTUAL_REF_SPADS                  = 0x00D9,
      REF_SPAD_CHAR_RESULT__REF_LOCATION                          = 0x00DA,
      VHV_RESULT__COLDBOOT_STATUS                                 = 0x00DB,
      VHV_RESULT__SEARCH_RESULT                                   = 0x00DC,
      VHV_RESULT__LATEST_SETTING                                  = 0x00DD,
      RESULT__OSC_CALIBRATE_VAL                                   = 0x00DE,
      RESULT__OSC_CALIBRATE_VAL_HI                                = 0x00DE,
      RESULT__OSC_CALIBRATE_VAL_LO                                = 0x00DF,
      ANA_CONFIG__POWERDOWN_GO1                                   = 0x00E0,
      ANA_CONFIG__REF_BG_CTRL                                     = 0x00E1,
      ANA_CONFIG__REGDVDD1V2_CTRL                                 = 0x00E2,
      ANA_CONFIG__OSC_SLOW_CTRL                                   = 0x00E3,
      TEST_MODE__STATUS                                           = 0x00E4,
      FIRMWARE__SYSTEM_STATUS                                     = 0x00E5,
      FIRMWARE__MODE_STATUS                                       = 0x00E6,
      FIRMWARE__SECONDARY_MODE_STATUS                             = 0x00E7,
      FIRMWARE__CAL_REPEAT_RATE_COUNTER                           = 0x00E8,
      FIRMWARE__CAL_REPEAT_RATE_COUNTER_HI                        = 0x00E8,
      FIRMWARE__CAL_REPEAT_RATE_COUNTER_LO                        = 0x00E9,
      FIRMWARE__HISTOGRAM_BIN                                     = 0x00EA,
      GPH__SYSTEM__THRESH_HIGH                                    = 0x00EC,
      GPH__SYSTEM__THRESH_HIGH_HI                                 = 0x00EC,
```

```
      GPH__SYSTEM__THRESH_HIGH_LO                                  = 0x00ED,
      GPH__SYSTEM__THRESH_LOW                                      = 0x00EE,
      GPH__SYSTEM__THRESH_LOW_HI                                   = 0x00EE,
      GPH__SYSTEM__THRESH_LOW_LO                                   = 0x00EF,
      GPH__SYSTEM__ENABLE_XTALK_PER_QUADRANT                       = 0x00F0,
      GPH__SPARE_0                                                 = 0x00F1,
      GPH__SD_CONFIG__WOI_SD0                                      = 0x00F2,
      GPH__SD_CONFIG__WOI_SD1                                      = 0x00F3,
      GPH__SD_CONFIG__INITIAL_PHASE_SD0                            = 0x00F4,
      GPH__SD_CONFIG__INITIAL_PHASE_SD1                            = 0x00F5,
      GPH__SD_CONFIG__FIRST_ORDER_SELECT                           = 0x00F6,
      GPH__SD_CONFIG__QUANTIFIER                                   = 0x00F7,
      GPH__ROI_CONFIG__USER_ROI_CENTRE_SPAD                        = 0x00F8,
      GPH__ROI_CONFIG__USER_ROI_REQUESTED_GLOBAL_XY_SIZE           = 0x00F9,
      GPH__SYSTEM__SEQUENCE_CONFIG                                 = 0x00FA,
      GPH__GPH_ID                                                  = 0x00FB,
      SYSTEM__INTERRUPT_SET                                        = 0x00FC,
      INTERRUPT_MANAGER__ENABLES                                   = 0x00FD,
      INTERRUPT_MANAGER__CLEAR                                     = 0x00FE,
      INTERRUPT_MANAGER__STATUS                                    = 0x00FF,
      MCU_TO_HOST_BANK__WR_ACCESS_EN                               = 0x0100,
      POWER_MANAGEMENT__GO1_RESET_STATUS                           = 0x0101,
      PAD_STARTUP_MODE__VALUE_RO                                   = 0x0102,
      PAD_STARTUP_MODE__VALUE_CTRL                                 = 0x0103,
      PLL_PERIOD_US                                                = 0x0104,
      PLL_PERIOD_US_3                                              = 0x0104,
      PLL_PERIOD_US_2                                              = 0x0105,
      PLL_PERIOD_US_1                                              = 0x0106,
      PLL_PERIOD_US_0                                              = 0x0107,
      INTERRUPT_SCHEDULER__DATA_OUT                                = 0x0108,
      INTERRUPT_SCHEDULER__DATA_OUT_3                              = 0x0108,
      INTERRUPT_SCHEDULER__DATA_OUT_2                              = 0x0109,
      INTERRUPT_SCHEDULER__DATA_OUT_1                              = 0x010A,
      INTERRUPT_SCHEDULER__DATA_OUT_0                              = 0x010B,
      NVM_BIST__COMPLETE                                           = 0x010C,
      NVM_BIST__STATUS                                             = 0x010D,
      IDENTIFICATION__MODEL_ID                                     = 0x010F,
      IDENTIFICATION__MODULE_TYPE                                  = 0x0110,
      IDENTIFICATION__REVISION_ID                                  = 0x0111,
      IDENTIFICATION__MODULE_ID                                    = 0x0112,
      IDENTIFICATION__MODULE_ID_HI                                 = 0x0112,
      IDENTIFICATION__MODULE_ID_LO                                 = 0x0113,
      ANA_CONFIG__FAST_OSC__TRIM_MAX                               = 0x0114,
      ANA_CONFIG__FAST_OSC__FREQ_SET                               = 0x0115,
      ANA_CONFIG__VCSEL_TRIM                                       = 0x0116,
      ANA_CONFIG__VCSEL_SELION                                     = 0x0117,
      ANA_CONFIG__VCSEL_SELION_MAX                                 = 0x0118,
      PROTECTED_LASER_SAFETY__LOCK_BIT                             = 0x0119,
      LASER_SAFETY__KEY                                            = 0x011A,
      LASER_SAFETY__KEY_RO                                         = 0x011B,
```

```
      LASER_SAFETY__CLIP                                                = 0x011C,
      LASER_SAFETY__MULT                                                = 0x011D,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_0                                 = 0x011E,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_1                                 = 0x011F,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_2                                 = 0x0120,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_3                                 = 0x0121,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_4                                 = 0x0122,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_5                                 = 0x0123,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_6                                 = 0x0124,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_7                                 = 0x0125,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_8                                 = 0x0126,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_9                                 = 0x0127,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_10                                = 0x0128,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_11                                = 0x0129,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_12                                = 0x012A,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_13                                = 0x012B,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_14                                = 0x012C,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_15                                = 0x012D,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_16                                = 0x012E,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_17                                = 0x012F,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_18                                = 0x0130,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_19                                = 0x0131,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_20                                = 0x0132,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_21                                = 0x0133,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_22                                = 0x0134,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_23                                = 0x0135,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_24                                = 0x0136,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_25                                = 0x0137,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_26                                = 0x0138,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_27                                = 0x0139,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_28                                = 0x013A,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_29                                = 0x013B,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_30                                = 0x013C,
      GLOBAL_CONFIG__SPAD_ENABLES_RTN_31                                = 0x013D,
      ROI_CONFIG__MODE_ROI_CENTRE_SPAD                                  = 0x013E,
      ROI_CONFIG__MODE_ROI_XY_SIZE                                      = 0x013F,
      GO2_HOST_BANK_ACCESS__OVERRIDE                                    = 0x0300,
      MCU_UTIL_MULTIPLIER__MULTIPLICAND                                 = 0x0400,
      MCU_UTIL_MULTIPLIER__MULTIPLICAND_3                               = 0x0400,
      MCU_UTIL_MULTIPLIER__MULTIPLICAND_2                               = 0x0401,
      MCU_UTIL_MULTIPLIER__MULTIPLICAND_1                               = 0x0402,
      MCU_UTIL_MULTIPLIER__MULTIPLICAND_0                               = 0x0403,
      MCU_UTIL_MULTIPLIER__MULTIPLIER                                   = 0x0404,
      MCU_UTIL_MULTIPLIER__MULTIPLIER_3                                 = 0x0404,
      MCU_UTIL_MULTIPLIER__MULTIPLIER_2                                 = 0x0405,
      MCU_UTIL_MULTIPLIER__MULTIPLIER_1                                 = 0x0406,
      MCU_UTIL_MULTIPLIER__MULTIPLIER_0                                 = 0x0407,
      MCU_UTIL_MULTIPLIER__PRODUCT_HI                                   = 0x0408,
      MCU_UTIL_MULTIPLIER__PRODUCT_HI_3                                 = 0x0408,
      MCU_UTIL_MULTIPLIER__PRODUCT_HI_2                                 = 0x0409,
```

```
    MCU_UTIL_MULTIPLIER__PRODUCT_HI_1                                = 0x040A,
    MCU_UTIL_MULTIPLIER__PRODUCT_HI_0                                = 0x040B,
    MCU_UTIL_MULTIPLIER__PRODUCT_LO                                  = 0x040C,
    MCU_UTIL_MULTIPLIER__PRODUCT_LO_3                                = 0x040C,
    MCU_UTIL_MULTIPLIER__PRODUCT_LO_2                                = 0x040D,
    MCU_UTIL_MULTIPLIER__PRODUCT_LO_1                                = 0x040E,
    MCU_UTIL_MULTIPLIER__PRODUCT_LO_0                                = 0x040F,
    MCU_UTIL_MULTIPLIER__START                                       = 0x0410,
    MCU_UTIL_MULTIPLIER__STATUS                                      = 0x0411,
    MCU_UTIL_DIVIDER__START                                          = 0x0412,
    MCU_UTIL_DIVIDER__STATUS                                         = 0x0413,
    MCU_UTIL_DIVIDER__DIVIDEND                                       = 0x0414,
    MCU_UTIL_DIVIDER__DIVIDEND_3                                     = 0x0414,
    MCU_UTIL_DIVIDER__DIVIDEND_2                                     = 0x0415,
    MCU_UTIL_DIVIDER__DIVIDEND_1                                     = 0x0416,
    MCU_UTIL_DIVIDER__DIVIDEND_0                                     = 0x0417,
    MCU_UTIL_DIVIDER__DIVISOR                                        = 0x0418,
    MCU_UTIL_DIVIDER__DIVISOR_3                                      = 0x0418,
    MCU_UTIL_DIVIDER__DIVISOR_2                                      = 0x0419,
    MCU_UTIL_DIVIDER__DIVISOR_1                                      = 0x041A,
    MCU_UTIL_DIVIDER__DIVISOR_0                                      = 0x041B,
    MCU_UTIL_DIVIDER__QUOTIENT                                       = 0x041C,
    MCU_UTIL_DIVIDER__QUOTIENT_3                                     = 0x041C,
    MCU_UTIL_DIVIDER__QUOTIENT_2                                     = 0x041D,
    MCU_UTIL_DIVIDER__QUOTIENT_1                                     = 0x041E,
    MCU_UTIL_DIVIDER__QUOTIENT_0                                     = 0x041F,
    TIMER0__VALUE_IN                                                 = 0x0420,
    TIMER0__VALUE_IN_3                                               = 0x0420,
    TIMER0__VALUE_IN_2                                               = 0x0421,
    TIMER0__VALUE_IN_1                                               = 0x0422,
    TIMER0__VALUE_IN_0                                               = 0x0423,
    TIMER1__VALUE_IN                                                 = 0x0424,
    TIMER1__VALUE_IN_3                                               = 0x0424,
    TIMER1__VALUE_IN_2                                               = 0x0425,
    TIMER1__VALUE_IN_1                                               = 0x0426,
    TIMER1__VALUE_IN_0                                               = 0x0427,
    TIMER0__CTRL                                                     = 0x0428,
    TIMER1__CTRL                                                     = 0x0429,
    MCU_GENERAL_PURPOSE__GP_0                                        = 0x042C,
    MCU_GENERAL_PURPOSE__GP_1                                        = 0x042D,
    MCU_GENERAL_PURPOSE__GP_2                                        = 0x042E,
    MCU_GENERAL_PURPOSE__GP_3                                        = 0x042F,
    MCU_RANGE_CALC__CONFIG                                           = 0x0430,
    MCU_RANGE_CALC__OFFSET_CORRECTED_RANGE                           = 0x0432,
    MCU_RANGE_CALC__OFFSET_CORRECTED_RANGE_HI                        = 0x0432,
    MCU_RANGE_CALC__OFFSET_CORRECTED_RANGE_LO                        = 0x0433,
    MCU_RANGE_CALC__SPARE_4                                          = 0x0434,
    MCU_RANGE_CALC__SPARE_4_3                                        = 0x0434,
    MCU_RANGE_CALC__SPARE_4_2                                        = 0x0435,
    MCU_RANGE_CALC__SPARE_4_1                                        = 0x0436,
```

```
    MCU_RANGE_CALC__SPARE_4_0                                = 0x0437,
    MCU_RANGE_CALC__AMBIENT_DURATION_PRE_CALC                = 0x0438,
    MCU_RANGE_CALC__AMBIENT_DURATION_PRE_CALC_HI             = 0x0438,
    MCU_RANGE_CALC__AMBIENT_DURATION_PRE_CALC_LO             = 0x0439,
    MCU_RANGE_CALC__ALGO_VCSEL_PERIOD                        = 0x043C,
    MCU_RANGE_CALC__SPARE_5                                  = 0x043D,
    MCU_RANGE_CALC__ALGO_TOTAL_PERIODS                       = 0x043E,
    MCU_RANGE_CALC__ALGO_TOTAL_PERIODS_HI                    = 0x043E,
    MCU_RANGE_CALC__ALGO_TOTAL_PERIODS_LO                    = 0x043F,
    MCU_RANGE_CALC__ALGO_ACCUM_PHASE                         = 0x0440,
    MCU_RANGE_CALC__ALGO_ACCUM_PHASE_3                       = 0x0440,
    MCU_RANGE_CALC__ALGO_ACCUM_PHASE_2                       = 0x0441,
    MCU_RANGE_CALC__ALGO_ACCUM_PHASE_1                       = 0x0442,
    MCU_RANGE_CALC__ALGO_ACCUM_PHASE_0                       = 0x0443,
    MCU_RANGE_CALC__ALGO_SIGNAL_EVENTS                       = 0x0444,
    MCU_RANGE_CALC__ALGO_SIGNAL_EVENTS_3                     = 0x0444,
    MCU_RANGE_CALC__ALGO_SIGNAL_EVENTS_2                     = 0x0445,
    MCU_RANGE_CALC__ALGO_SIGNAL_EVENTS_1                     = 0x0446,
    MCU_RANGE_CALC__ALGO_SIGNAL_EVENTS_0                     = 0x0447,
    MCU_RANGE_CALC__ALGO_AMBIENT_EVENTS                      = 0x0448,
    MCU_RANGE_CALC__ALGO_AMBIENT_EVENTS_3                    = 0x0448,
    MCU_RANGE_CALC__ALGO_AMBIENT_EVENTS_2                    = 0x0449,
    MCU_RANGE_CALC__ALGO_AMBIENT_EVENTS_1                    = 0x044A,
    MCU_RANGE_CALC__ALGO_AMBIENT_EVENTS_0                    = 0x044B,
    MCU_RANGE_CALC__SPARE_6                                  = 0x044C,
    MCU_RANGE_CALC__SPARE_6_HI                               = 0x044C,
    MCU_RANGE_CALC__SPARE_6_LO                               = 0x044D,
    MCU_RANGE_CALC__ALGO_ADJUST_VCSEL_PERIOD                 = 0x044E,
    MCU_RANGE_CALC__ALGO_ADJUST_VCSEL_PERIOD_HI              = 0x044E,
    MCU_RANGE_CALC__ALGO_ADJUST_VCSEL_PERIOD_LO              = 0x044F,
    MCU_RANGE_CALC__NUM_SPADS                                = 0x0450,
    MCU_RANGE_CALC__NUM_SPADS_HI                             = 0x0450,
    MCU_RANGE_CALC__NUM_SPADS_LO                             = 0x0451,
    MCU_RANGE_CALC__PHASE_OUTPUT                             = 0x0452,
    MCU_RANGE_CALC__PHASE_OUTPUT_HI                          = 0x0452,
    MCU_RANGE_CALC__PHASE_OUTPUT_LO                          = 0x0453,
    MCU_RANGE_CALC__RATE_PER_SPAD_MCPS                       = 0x0454,
    MCU_RANGE_CALC__RATE_PER_SPAD_MCPS_3                     = 0x0454,
    MCU_RANGE_CALC__RATE_PER_SPAD_MCPS_2                     = 0x0455,
    MCU_RANGE_CALC__RATE_PER_SPAD_MCPS_1                     = 0x0456,
    MCU_RANGE_CALC__RATE_PER_SPAD_MCPS_0                     = 0x0457,
    MCU_RANGE_CALC__SPARE_7                                  = 0x0458,
    MCU_RANGE_CALC__SPARE_8                                  = 0x0459,
    MCU_RANGE_CALC__PEAK_SIGNAL_RATE_MCPS                    = 0x045A,
    MCU_RANGE_CALC__PEAK_SIGNAL_RATE_MCPS_HI                 = 0x045A,
    MCU_RANGE_CALC__PEAK_SIGNAL_RATE_MCPS_LO                 = 0x045B,
    MCU_RANGE_CALC__AVG_SIGNAL_RATE_MCPS                     = 0x045C,
    MCU_RANGE_CALC__AVG_SIGNAL_RATE_MCPS_HI                  = 0x045C,
    MCU_RANGE_CALC__AVG_SIGNAL_RATE_MCPS_LO                  = 0x045D,
    MCU_RANGE_CALC__AMBIENT_RATE_MCPS                        = 0x045E,
```

```
    MCU_RANGE_CALC__AMBIENT_RATE_MCPS_HI                                = 0x045E,
    MCU_RANGE_CALC__AMBIENT_RATE_MCPS_LO                                = 0x045F,
    MCU_RANGE_CALC__XTALK                                               = 0x0460,
    MCU_RANGE_CALC__XTALK_HI                                            = 0x0460,
    MCU_RANGE_CALC__XTALK_LO                                            = 0x0461,
    MCU_RANGE_CALC__CALC_STATUS                                         = 0x0462,
    MCU_RANGE_CALC__DEBUG                                               = 0x0463,
    MCU_RANGE_CALC__PEAK_SIGNAL_RATE_XTALK_CORR_MCPS                    = 0x0464,
    MCU_RANGE_CALC__PEAK_SIGNAL_RATE_XTALK_CORR_MCPS_HI                 = 0x0464,
    MCU_RANGE_CALC__PEAK_SIGNAL_RATE_XTALK_CORR_MCPS_LO                 = 0x0465,
    MCU_RANGE_CALC__SPARE_0                                             = 0x0468,
    MCU_RANGE_CALC__SPARE_1                                             = 0x0469,
    MCU_RANGE_CALC__SPARE_2                                             = 0x046A,
    MCU_RANGE_CALC__SPARE_3                                             = 0x046B,
    PATCH__CTRL                                                         = 0x0470,
    PATCH__JMP_ENABLES                                                  = 0x0472,
    PATCH__JMP_ENABLES_HI                                               = 0x0472,
    PATCH__JMP_ENABLES_LO                                               = 0x0473,
    PATCH__DATA_ENABLES                                                 = 0x0474,
    PATCH__DATA_ENABLES_HI                                              = 0x0474,
    PATCH__DATA_ENABLES_LO                                              = 0x0475,
    PATCH__OFFSET_0                                                     = 0x0476,
    PATCH__OFFSET_0_HI                                                  = 0x0476,
    PATCH__OFFSET_0_LO                                                  = 0x0477,
    PATCH__OFFSET_1                                                     = 0x0478,
    PATCH__OFFSET_1_HI                                                  = 0x0478,
    PATCH__OFFSET_1_LO                                                  = 0x0479,
    PATCH__OFFSET_2                                                     = 0x047A,
    PATCH__OFFSET_2_HI                                                  = 0x047A,
    PATCH__OFFSET_2_LO                                                  = 0x047B,
    PATCH__OFFSET_3                                                     = 0x047C,
    PATCH__OFFSET_3_HI                                                  = 0x047C,
    PATCH__OFFSET_3_LO                                                  = 0x047D,
    PATCH__OFFSET_4                                                     = 0x047E,
    PATCH__OFFSET_4_HI                                                  = 0x047E,
    PATCH__OFFSET_4_LO                                                  = 0x047F,
    PATCH__OFFSET_5                                                     = 0x0480,
    PATCH__OFFSET_5_HI                                                  = 0x0480,
    PATCH__OFFSET_5_LO                                                  = 0x0481,
    PATCH__OFFSET_6                                                     = 0x0482,
    PATCH__OFFSET_6_HI                                                  = 0x0482,
    PATCH__OFFSET_6_LO                                                  = 0x0483,
    PATCH__OFFSET_7                                                     = 0x0484,
    PATCH__OFFSET_7_HI                                                  = 0x0484,
    PATCH__OFFSET_7_LO                                                  = 0x0485,
    PATCH__OFFSET_8                                                     = 0x0486,
    PATCH__OFFSET_8_HI                                                  = 0x0486,
    PATCH__OFFSET_8_LO                                                  = 0x0487,
    PATCH__OFFSET_9                                                     = 0x0488,
    PATCH__OFFSET_9_HI                                                  = 0x0488,
```

```
    PATCH__OFFSET_9_LO                                      = 0x0489,
    PATCH__OFFSET_10                                        = 0x048A,
    PATCH__OFFSET_10_HI                                     = 0x048A,
    PATCH__OFFSET_10_LO                                     = 0x048B,
    PATCH__OFFSET_11                                        = 0x048C,
    PATCH__OFFSET_11_HI                                     = 0x048C,
    PATCH__OFFSET_11_LO                                     = 0x048D,
    PATCH__OFFSET_12                                        = 0x048E,
    PATCH__OFFSET_12_HI                                     = 0x048E,
    PATCH__OFFSET_12_LO                                     = 0x048F,
    PATCH__OFFSET_13                                        = 0x0490,
    PATCH__OFFSET_13_HI                                     = 0x0490,
    PATCH__OFFSET_13_LO                                     = 0x0491,
    PATCH__OFFSET_14                                        = 0x0492,
    PATCH__OFFSET_14_HI                                     = 0x0492,
    PATCH__OFFSET_14_LO                                     = 0x0493,
    PATCH__OFFSET_15                                        = 0x0494,
    PATCH__OFFSET_15_HI                                     = 0x0494,
    PATCH__OFFSET_15_LO                                     = 0x0495,
    PATCH__ADDRESS_0                                        = 0x0496,
    PATCH__ADDRESS_0_HI                                     = 0x0496,
    PATCH__ADDRESS_0_LO                                     = 0x0497,
    PATCH__ADDRESS_1                                        = 0x0498,
    PATCH__ADDRESS_1_HI                                     = 0x0498,
    PATCH__ADDRESS_1_LO                                     = 0x0499,
    PATCH__ADDRESS_2                                        = 0x049A,
    PATCH__ADDRESS_2_HI                                     = 0x049A,
    PATCH__ADDRESS_2_LO                                     = 0x049B,
    PATCH__ADDRESS_3                                        = 0x049C,
    PATCH__ADDRESS_3_HI                                     = 0x049C,
    PATCH__ADDRESS_3_LO                                     = 0x049D,
    PATCH__ADDRESS_4                                        = 0x049E,
    PATCH__ADDRESS_4_HI                                     = 0x049E,
    PATCH__ADDRESS_4_LO                                     = 0x049F,
    PATCH__ADDRESS_5                                        = 0x04A0,
    PATCH__ADDRESS_5_HI                                     = 0x04A0,
    PATCH__ADDRESS_5_LO                                     = 0x04A1,
    PATCH__ADDRESS_6                                        = 0x04A2,
    PATCH__ADDRESS_6_HI                                     = 0x04A2,
    PATCH__ADDRESS_6_LO                                     = 0x04A3,
    PATCH__ADDRESS_7                                        = 0x04A4,
    PATCH__ADDRESS_7_HI                                     = 0x04A4,
    PATCH__ADDRESS_7_LO                                     = 0x04A5,
    PATCH__ADDRESS_8                                        = 0x04A6,
    PATCH__ADDRESS_8_HI                                     = 0x04A6,
    PATCH__ADDRESS_8_LO                                     = 0x04A7,
    PATCH__ADDRESS_9                                        = 0x04A8,
    PATCH__ADDRESS_9_HI                                     = 0x04A8,
    PATCH__ADDRESS_9_LO                                     = 0x04A9,
    PATCH__ADDRESS_10                                       = 0x04AA,
```

```
    PATCH__ADDRESS_10_HI                                    = 0x04AA,
    PATCH__ADDRESS_10_LO                                    = 0x04AB,
    PATCH__ADDRESS_11                                       = 0x04AC,
    PATCH__ADDRESS_11_HI                                    = 0x04AC,
    PATCH__ADDRESS_11_LO                                    = 0x04AD,
    PATCH__ADDRESS_12                                       = 0x04AE,
    PATCH__ADDRESS_12_HI                                    = 0x04AE,
    PATCH__ADDRESS_12_LO                                    = 0x04AF,
    PATCH__ADDRESS_13                                       = 0x04B0,
    PATCH__ADDRESS_13_HI                                    = 0x04B0,
    PATCH__ADDRESS_13_LO                                    = 0x04B1,
    PATCH__ADDRESS_14                                       = 0x04B2,
    PATCH__ADDRESS_14_HI                                    = 0x04B2,
    PATCH__ADDRESS_14_LO                                    = 0x04B3,
    PATCH__ADDRESS_15                                       = 0x04B4,
    PATCH__ADDRESS_15_HI                                    = 0x04B4,
    PATCH__ADDRESS_15_LO                                    = 0x04B5,
    SPI_ASYNC_MUX__CTRL                                     = 0x04C0,
    CLK__CONFIG                                             = 0x04C4,
    GPIO_LV_MUX__CTRL                                       = 0x04CC,
    GPIO_LV_PAD__CTRL                                       = 0x04CD,
    PAD_I2C_LV__CONFIG                                      = 0x04D0,
    PAD_STARTUP_MODE__VALUE_RO_GO1                          = 0x04D4,
    HOST_IF__STATUS_GO1                                     = 0x04D5,
    MCU_CLK_GATING__CTRL                                    = 0x04D8,
    TEST__BIST_ROM_CTRL                                     = 0x04E0,
    TEST__BIST_ROM_RESULT                                   = 0x04E1,
    TEST__BIST_ROM_MCU_SIG                                  = 0x04E2,
    TEST__BIST_ROM_MCU_SIG_HI                               = 0x04E2,
    TEST__BIST_ROM_MCU_SIG_LO                               = 0x04E3,
    TEST__BIST_RAM_CTRL                                     = 0x04E4,
    TEST__BIST_RAM_RESULT                                   = 0x04E5,
    TEST__TMC                                               = 0x04E8,
    TEST__PLL_BIST_MIN_THRESHOLD                            = 0x04F0,
    TEST__PLL_BIST_MIN_THRESHOLD_HI                         = 0x04F0,
    TEST__PLL_BIST_MIN_THRESHOLD_LO                         = 0x04F1,
    TEST__PLL_BIST_MAX_THRESHOLD                            = 0x04F2,
    TEST__PLL_BIST_MAX_THRESHOLD_HI                         = 0x04F2,
    TEST__PLL_BIST_MAX_THRESHOLD_LO                         = 0x04F3,
    TEST__PLL_BIST_COUNT_OUT                                = 0x04F4,
    TEST__PLL_BIST_COUNT_OUT_HI                             = 0x04F4,
    TEST__PLL_BIST_COUNT_OUT_LO                             = 0x04F5,
    TEST__PLL_BIST_GONOGO                                   = 0x04F6,
    TEST__PLL_BIST_CTRL                                     = 0x04F7,
    RANGING_CORE__DEVICE_ID                                 = 0x0680,
    RANGING_CORE__REVISION_ID                               = 0x0681,
    RANGING_CORE__CLK_CTRL1                                 = 0x0683,
    RANGING_CORE__CLK_CTRL2                                 = 0x0684,
    RANGING_CORE__WOI_1                                     = 0x0685,
    RANGING_CORE__WOI_REF_1                                 = 0x0686,
```

```
    RANGING_CORE__START_RANGING                                 = 0x0687,
    RANGING_CORE__LOW_LIMIT_1                                   = 0x0690,
    RANGING_CORE__HIGH_LIMIT_1                                  = 0x0691,
    RANGING_CORE__LOW_LIMIT_REF_1                               = 0x0692,
    RANGING_CORE__HIGH_LIMIT_REF_1                              = 0x0693,
    RANGING_CORE__QUANTIFIER_1_MSB                              = 0x0694,
    RANGING_CORE__QUANTIFIER_1_LSB                              = 0x0695,
    RANGING_CORE__QUANTIFIER_REF_1_MSB                          = 0x0696,
    RANGING_CORE__QUANTIFIER_REF_1_LSB                          = 0x0697,
    RANGING_CORE__AMBIENT_OFFSET_1_MSB                          = 0x0698,
    RANGING_CORE__AMBIENT_OFFSET_1_LSB                          = 0x0699,
    RANGING_CORE__AMBIENT_OFFSET_REF_1_MSB                      = 0x069A,
    RANGING_CORE__AMBIENT_OFFSET_REF_1_LSB                      = 0x069B,
    RANGING_CORE__FILTER_STRENGTH_1                             = 0x069C,
    RANGING_CORE__FILTER_STRENGTH_REF_1                         = 0x069D,
    RANGING_CORE__SIGNAL_EVENT_LIMIT_1_MSB                      = 0x069E,
    RANGING_CORE__SIGNAL_EVENT_LIMIT_1_LSB                      = 0x069F,
    RANGING_CORE__SIGNAL_EVENT_LIMIT_REF_1_MSB                  = 0x06A0,
    RANGING_CORE__SIGNAL_EVENT_LIMIT_REF_1_LSB                  = 0x06A1,
    RANGING_CORE__TIMEOUT_OVERALL_PERIODS_MSB                   = 0x06A4,
    RANGING_CORE__TIMEOUT_OVERALL_PERIODS_LSB                   = 0x06A5,
    RANGING_CORE__INVERT_HW                                     = 0x06A6,
    RANGING_CORE__FORCE_HW                                      = 0x06A7,
    RANGING_CORE__STATIC_HW_VALUE                               = 0x06A8,
    RANGING_CORE__FORCE_CONTINUOUS_AMBIENT                      = 0x06A9,
    RANGING_CORE__TEST_PHASE_SELECT_TO_FILTER                   = 0x06AA,
    RANGING_CORE__TEST_PHASE_SELECT_TO_TIMING_GEN               = 0x06AB,
    RANGING_CORE__INITIAL_PHASE_VALUE_1                         = 0x06AC,
    RANGING_CORE__INITIAL_PHASE_VALUE_REF_1                     = 0x06AD,
    RANGING_CORE__FORCE_UP_IN                                   = 0x06AE,
    RANGING_CORE__FORCE_DN_IN                                   = 0x06AF,
    RANGING_CORE__STATIC_UP_VALUE_1                             = 0x06B0,
    RANGING_CORE__STATIC_UP_VALUE_REF_1                         = 0x06B1,
    RANGING_CORE__STATIC_DN_VALUE_1                             = 0x06B2,
    RANGING_CORE__STATIC_DN_VALUE_REF_1                         = 0x06B3,
    RANGING_CORE__MONITOR_UP_DN                                 = 0x06B4,
    RANGING_CORE__INVERT_UP_DN                                  = 0x06B5,
    RANGING_CORE__CPUMP_1                                       = 0x06B6,
    RANGING_CORE__CPUMP_2                                       = 0x06B7,
    RANGING_CORE__CPUMP_3                                       = 0x06B8,
    RANGING_CORE__OSC_1                                         = 0x06B9,
    RANGING_CORE__PLL_1                                         = 0x06BB,
    RANGING_CORE__PLL_2                                         = 0x06BC,
    RANGING_CORE__REFERENCE_1                                   = 0x06BD,
    RANGING_CORE__REFERENCE_3                                   = 0x06BF,
    RANGING_CORE__REFERENCE_4                                   = 0x06C0,
    RANGING_CORE__REFERENCE_5                                   = 0x06C1,
    RANGING_CORE__REGAVDD1V2                                    = 0x06C3,
    RANGING_CORE__CALIB_1                                       = 0x06C4,
    RANGING_CORE__CALIB_2                                       = 0x06C5,
```

```
      RANGING_CORE__CALIB_3                                          = 0x06C6,
      RANGING_CORE__TST_MUX_SEL1                                     = 0x06C9,
      RANGING_CORE__TST_MUX_SEL2                                     = 0x06CA,
      RANGING_CORE__TST_MUX                                          = 0x06CB,
      RANGING_CORE__GPIO_OUT_TESTMUX                                 = 0x06CC,
      RANGING_CORE__CUSTOM_FE                                        = 0x06CD,
      RANGING_CORE__CUSTOM_FE_2                                      = 0x06CE,
      RANGING_CORE__SPAD_READOUT                                     = 0x06CF,
      RANGING_CORE__SPAD_READOUT_1                                   = 0x06D0,
      RANGING_CORE__SPAD_READOUT_2                                   = 0x06D1,
      RANGING_CORE__SPAD_PS                                          = 0x06D2,
      RANGING_CORE__LASER_SAFETY_2                                   = 0x06D4,
      RANGING_CORE__NVM_CTRL__MODE                                   = 0x0780,
      RANGING_CORE__NVM_CTRL__PDN                                    = 0x0781,
      RANGING_CORE__NVM_CTRL__PROGN                                  = 0x0782,
      RANGING_CORE__NVM_CTRL__READN                                  = 0x0783,
      RANGING_CORE__NVM_CTRL__PULSE_WIDTH_MSB                        = 0x0784,
      RANGING_CORE__NVM_CTRL__PULSE_WIDTH_LSB                        = 0x0785,
      RANGING_CORE__NVM_CTRL__HV_RISE_MSB                            = 0x0786,
      RANGING_CORE__NVM_CTRL__HV_RISE_LSB                            = 0x0787,
      RANGING_CORE__NVM_CTRL__HV_FALL_MSB                            = 0x0788,
      RANGING_CORE__NVM_CTRL__HV_FALL_LSB                            = 0x0789,
      RANGING_CORE__NVM_CTRL__TST                                    = 0x078A,
      RANGING_CORE__NVM_CTRL__TESTREAD                               = 0x078B,
      RANGING_CORE__NVM_CTRL__DATAIN_MMM                             = 0x078C,
      RANGING_CORE__NVM_CTRL__DATAIN_LMM                             = 0x078D,
      RANGING_CORE__NVM_CTRL__DATAIN_LLM                             = 0x078E,
      RANGING_CORE__NVM_CTRL__DATAIN_LLL                             = 0x078F,
      RANGING_CORE__NVM_CTRL__DATAOUT_MMM                            = 0x0790,
      RANGING_CORE__NVM_CTRL__DATAOUT_LMM                            = 0x0791,
      RANGING_CORE__NVM_CTRL__DATAOUT_LLM                            = 0x0792,
      RANGING_CORE__NVM_CTRL__DATAOUT_LLL                            = 0x0793,
      RANGING_CORE__NVM_CTRL__ADDR                                   = 0x0794,
      RANGING_CORE__NVM_CTRL__DATAOUT_ECC                            = 0x0795,
      RANGING_CORE__RET_SPAD_EN_0                                    = 0x0796,
      RANGING_CORE__RET_SPAD_EN_1                                    = 0x0797,
      RANGING_CORE__RET_SPAD_EN_2                                    = 0x0798,
      RANGING_CORE__RET_SPAD_EN_3                                    = 0x0799,
      RANGING_CORE__RET_SPAD_EN_4                                    = 0x079A,
      RANGING_CORE__RET_SPAD_EN_5                                    = 0x079B,
      RANGING_CORE__RET_SPAD_EN_6                                    = 0x079C,
      RANGING_CORE__RET_SPAD_EN_7                                    = 0x079D,
      RANGING_CORE__RET_SPAD_EN_8                                    = 0x079E,
      RANGING_CORE__RET_SPAD_EN_9                                    = 0x079F,
      RANGING_CORE__RET_SPAD_EN_10                                   = 0x07A0,
      RANGING_CORE__RET_SPAD_EN_11                                   = 0x07A1,
      RANGING_CORE__RET_SPAD_EN_12                                   = 0x07A2,
      RANGING_CORE__RET_SPAD_EN_13                                   = 0x07A3,
      RANGING_CORE__RET_SPAD_EN_14                                   = 0x07A4,
      RANGING_CORE__RET_SPAD_EN_15                                   = 0x07A5,
```

```
      RANGING_CORE__RET_SPAD_EN_16                                = 0x07A6,
      RANGING_CORE__RET_SPAD_EN_17                                = 0x07A7,
      RANGING_CORE__SPAD_SHIFT_EN                                 = 0x07BA,
      RANGING_CORE__SPAD_DISABLE_CTRL                             = 0x07BB,
      RANGING_CORE__SPAD_EN_SHIFT_OUT_DEBUG                       = 0x07BC,
      RANGING_CORE__SPI_MODE                                      = 0x07BD,
      RANGING_CORE__GPIO_DIR                                      = 0x07BE,
      RANGING_CORE__VCSEL_PERIOD                                  = 0x0880,
      RANGING_CORE__VCSEL_START                                   = 0x0881,
      RANGING_CORE__VCSEL_STOP                                    = 0x0882,
      RANGING_CORE__VCSEL_1                                       = 0x0885,
      RANGING_CORE__VCSEL_STATUS                                  = 0x088D,
      RANGING_CORE__STATUS                                        = 0x0980,
      RANGING_CORE__LASER_CONTINUITY_STATE                        = 0x0981,
      RANGING_CORE__RANGE_1_MMM                                   = 0x0982,
      RANGING_CORE__RANGE_1_LMM                                   = 0x0983,
      RANGING_CORE__RANGE_1_LLM                                   = 0x0984,
      RANGING_CORE__RANGE_1_LLL                                   = 0x0985,
      RANGING_CORE__RANGE_REF_1_MMM                               = 0x0986,
      RANGING_CORE__RANGE_REF_1_LMM                               = 0x0987,
      RANGING_CORE__RANGE_REF_1_LLM                               = 0x0988,
      RANGING_CORE__RANGE_REF_1_LLL                               = 0x0989,
      RANGING_CORE__AMBIENT_WINDOW_EVENTS_1_MMM                   = 0x098A,
      RANGING_CORE__AMBIENT_WINDOW_EVENTS_1_LMM                   = 0x098B,
      RANGING_CORE__AMBIENT_WINDOW_EVENTS_1_LLM                   = 0x098C,
      RANGING_CORE__AMBIENT_WINDOW_EVENTS_1_LLL                   = 0x098D,
      RANGING_CORE__RANGING_TOTAL_EVENTS_1_MMM                    = 0x098E,
      RANGING_CORE__RANGING_TOTAL_EVENTS_1_LMM                    = 0x098F,
      RANGING_CORE__RANGING_TOTAL_EVENTS_1_LLM                    = 0x0990,
      RANGING_CORE__RANGING_TOTAL_EVENTS_1_LLL                    = 0x0991,
      RANGING_CORE__SIGNAL_TOTAL_EVENTS_1_MMM                     = 0x0992,
      RANGING_CORE__SIGNAL_TOTAL_EVENTS_1_LMM                     = 0x0993,
      RANGING_CORE__SIGNAL_TOTAL_EVENTS_1_LLM                     = 0x0994,
      RANGING_CORE__SIGNAL_TOTAL_EVENTS_1_LLL                     = 0x0995,
      RANGING_CORE__TOTAL_PERIODS_ELAPSED_1_MM                    = 0x0996,
      RANGING_CORE__TOTAL_PERIODS_ELAPSED_1_LM                    = 0x0997,
      RANGING_CORE__TOTAL_PERIODS_ELAPSED_1_LL                    = 0x0998,
      RANGING_CORE__AMBIENT_MISMATCH_MM                           = 0x0999,
      RANGING_CORE__AMBIENT_MISMATCH_LM                           = 0x099A,
      RANGING_CORE__AMBIENT_MISMATCH_LL                           = 0x099B,
      RANGING_CORE__AMBIENT_WINDOW_EVENTS_REF_1_MMM               = 0x099C,
      RANGING_CORE__AMBIENT_WINDOW_EVENTS_REF_1_LMM               = 0x099D,
      RANGING_CORE__AMBIENT_WINDOW_EVENTS_REF_1_LLM               = 0x099E,
      RANGING_CORE__AMBIENT_WINDOW_EVENTS_REF_1_LLL               = 0x099F,
      RANGING_CORE__RANGING_TOTAL_EVENTS_REF_1_MMM                = 0x09A0,
      RANGING_CORE__RANGING_TOTAL_EVENTS_REF_1_LMM                = 0x09A1,
      RANGING_CORE__RANGING_TOTAL_EVENTS_REF_1_LLM                = 0x09A2,
      RANGING_CORE__RANGING_TOTAL_EVENTS_REF_1_LLL                = 0x09A3,
      RANGING_CORE__SIGNAL_TOTAL_EVENTS_REF_1_MMM                 = 0x09A4,
      RANGING_CORE__SIGNAL_TOTAL_EVENTS_REF_1_LMM                 = 0x09A5,
```

```
    RANGING_CORE__SIGNAL_TOTAL_EVENTS_REF_1_LLM                 = 0x09A6,
    RANGING_CORE__SIGNAL_TOTAL_EVENTS_REF_1_LLL                 = 0x09A7,
    RANGING_CORE__TOTAL_PERIODS_ELAPSED_REF_1_MM                = 0x09A8,
    RANGING_CORE__TOTAL_PERIODS_ELAPSED_REF_1_LM                = 0x09A9,
    RANGING_CORE__TOTAL_PERIODS_ELAPSED_REF_1_LL                = 0x09AA,
    RANGING_CORE__AMBIENT_MISMATCH_REF_MM                       = 0x09AB,
    RANGING_CORE__AMBIENT_MISMATCH_REF_LM                       = 0x09AC,
    RANGING_CORE__AMBIENT_MISMATCH_REF_LL                       = 0x09AD,
    RANGING_CORE__GPIO_CONFIG__A0                               = 0x0A00,
    RANGING_CORE__RESET_CONTROL__A0                             = 0x0A01,
    RANGING_CORE__INTR_MANAGER__A0                              = 0x0A02,
    RANGING_CORE__POWER_FSM_TIME_OSC__A0                        = 0x0A06,
    RANGING_CORE__VCSEL_ATEST__A0                               = 0x0A07,
    RANGING_CORE__VCSEL_PERIOD_CLIPPED__A0                      = 0x0A08,
    RANGING_CORE__VCSEL_STOP_CLIPPED__A0                        = 0x0A09,
    RANGING_CORE__CALIB_2__A0                                   = 0x0A0A,
    RANGING_CORE__STOP_CONDITION__A0                            = 0x0A0B,
    RANGING_CORE__STATUS_RESET__A0                              = 0x0A0C,
    RANGING_CORE__READOUT_CFG__A0                               = 0x0A0D,
    RANGING_CORE__WINDOW_SETTING__A0                            = 0x0A0E,
    RANGING_CORE__VCSEL_DELAY__A0                               = 0x0A1A,
    RANGING_CORE__REFERENCE_2__A0                               = 0x0A1B,
    RANGING_CORE__REGAVDD1V2__A0                                = 0x0A1D,
    RANGING_CORE__TST_MUX__A0                                   = 0x0A1F,
    RANGING_CORE__CUSTOM_FE_2__A0                               = 0x0A20,
    RANGING_CORE__SPAD_READOUT__A0                              = 0x0A21,
    RANGING_CORE__CPUMP_1__A0                                   = 0x0A22,
    RANGING_CORE__SPARE_REGISTER__A0                            = 0x0A23,
    RANGING_CORE__VCSEL_CONT_STAGE5_BYPASS__A0                  = 0x0A24,
    RANGING_CORE__RET_SPAD_EN_18                                = 0x0A25,
    RANGING_CORE__RET_SPAD_EN_19                                = 0x0A26,
    RANGING_CORE__RET_SPAD_EN_20                                = 0x0A27,
    RANGING_CORE__RET_SPAD_EN_21                                = 0x0A28,
    RANGING_CORE__RET_SPAD_EN_22                                = 0x0A29,
    RANGING_CORE__RET_SPAD_EN_23                                = 0x0A2A,
    RANGING_CORE__RET_SPAD_EN_24                                = 0x0A2B,
    RANGING_CORE__RET_SPAD_EN_25                                = 0x0A2C,
    RANGING_CORE__RET_SPAD_EN_26                                = 0x0A2D,
    RANGING_CORE__RET_SPAD_EN_27                                = 0x0A2E,
    RANGING_CORE__RET_SPAD_EN_28                                = 0x0A2F,
    RANGING_CORE__RET_SPAD_EN_29                                = 0x0A30,
    RANGING_CORE__RET_SPAD_EN_30                                = 0x0A31,
    RANGING_CORE__RET_SPAD_EN_31                                = 0x0A32,
    RANGING_CORE__REF_SPAD_EN_0__EWOK                           = 0x0A33,
    RANGING_CORE__REF_SPAD_EN_1__EWOK                           = 0x0A34,
    RANGING_CORE__REF_SPAD_EN_2__EWOK                           = 0x0A35,
    RANGING_CORE__REF_SPAD_EN_3__EWOK                           = 0x0A36,
    RANGING_CORE__REF_SPAD_EN_4__EWOK                           = 0x0A37,
    RANGING_CORE__REF_SPAD_EN_5__EWOK                           = 0x0A38,
    RANGING_CORE__REF_EN_START_SELECT                           = 0x0A39,
```

```
    RANGING_CORE__REGDVDD1V2_ATEST__EWOK                                = 0x0A41,
    SOFT_RESET_GO1                                                      = 0x0B00,
    PRIVATE__PATCH_BASE_ADDR_RSLV                                       = 0x0E00,
    PREV_SHADOW_RESULT__INTERRUPT_STATUS                                = 0x0ED0,
    PREV_SHADOW_RESULT__RANGE_STATUS                                    = 0x0ED1,
    PREV_SHADOW_RESULT__REPORT_STATUS                                   = 0x0ED2,
    PREV_SHADOW_RESULT__STREAM_COUNT                                    = 0x0ED3,
    PREV_SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD0                  = 0x0ED4,
    PREV_SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD0_HI               = 0x0ED4,
    PREV_SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD0_LO               = 0x0ED5,
    PREV_SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD0                 = 0x0ED6,
    PREV_SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD0_HI              = 0x0ED6,
    PREV_SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD0_LO              = 0x0ED7,
    PREV_SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD0                     = 0x0ED8,
    PREV_SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD0_HI                  = 0x0ED8,
    PREV_SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD0_LO                  = 0x0ED9,
    PREV_SHADOW_RESULT__SIGMA_SD0                                       = 0x0EDA,
    PREV_SHADOW_RESULT__SIGMA_SD0_HI                                    = 0x0EDA,
    PREV_SHADOW_RESULT__SIGMA_SD0_LO                                    = 0x0EDB,
    PREV_SHADOW_RESULT__PHASE_SD0                                       = 0x0EDC,
    PREV_SHADOW_RESULT__PHASE_SD0_HI                                    = 0x0EDC,
    PREV_SHADOW_RESULT__PHASE_SD0_LO                                    = 0x0EDD,
    PREV_SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD0          = 0x0EDE,
    PREV_SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD0_HI       = 0x0EDE,
    PREV_SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD0_LO       = 0x0EDF,
    PREV_SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_CROSSTALK_CORRECTED_MCPS_SD0    = 0x0EE0,
    PREV_SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_CROSSTALK_CORRECTED_MCPS_SD0_HI = 0x0EE0,
    PREV_SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_CROSSTALK_CORRECTED_MCPS_SD0_LO = 0x0EE1,
    PREV_SHADOW_RESULT__MM_INNER_ACTUAL_EFFECTIVE_SPADS_SD0             = 0x0EE2,
    PREV_SHADOW_RESULT__MM_INNER_ACTUAL_EFFECTIVE_SPADS_SD0_HI          = 0x0EE2,
    PREV_SHADOW_RESULT__MM_INNER_ACTUAL_EFFECTIVE_SPADS_SD0_LO          = 0x0EE3,
    PREV_SHADOW_RESULT__MM_OUTER_ACTUAL_EFFECTIVE_SPADS_SD0             = 0x0EE4,
    PREV_SHADOW_RESULT__MM_OUTER_ACTUAL_EFFECTIVE_SPADS_SD0_HI          = 0x0EE4,
    PREV_SHADOW_RESULT__MM_OUTER_ACTUAL_EFFECTIVE_SPADS_SD0_LO          = 0x0EE5,
    PREV_SHADOW_RESULT__AVG_SIGNAL_COUNT_RATE_MCPS_SD0                  = 0x0EE6,
    PREV_SHADOW_RESULT__AVG_SIGNAL_COUNT_RATE_MCPS_SD0_HI               = 0x0EE6,
    PREV_SHADOW_RESULT__AVG_SIGNAL_COUNT_RATE_MCPS_SD0_LO               = 0x0EE7,
    PREV_SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD1                  = 0x0EE8,
    PREV_SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD1_HI               = 0x0EE8,
    PREV_SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD1_LO               = 0x0EE9,
    PREV_SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD1                 = 0x0EEA,
    PREV_SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD1_HI              = 0x0EEA,
    PREV_SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD1_LO              = 0x0EEB,
    PREV_SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD1                     = 0x0EEC,
    PREV_SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD1_HI                  = 0x0EEC,
    PREV_SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD1_LO                  = 0x0EED,
    PREV_SHADOW_RESULT__SIGMA_SD1                                       = 0x0EEE,
    PREV_SHADOW_RESULT__SIGMA_SD1_HI                                    = 0x0EEE,
    PREV_SHADOW_RESULT__SIGMA_SD1_LO                                    = 0x0EEF,
    PREV_SHADOW_RESULT__PHASE_SD1                                       = 0x0EF0,
```

```
    PREV_SHADOW_RESULT__PHASE_SD1_HI                                = 0x0EF0,
    PREV_SHADOW_RESULT__PHASE_SD1_LO                                = 0x0EF1,
    PREV_SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD1      = 0x0EF2,
    PREV_SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD1_HI   = 0x0EF2,
    PREV_SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD1_LO   = 0x0EF3,
    PREV_SHADOW_RESULT__SPARE_0_SD1                                 = 0x0EF4,
    PREV_SHADOW_RESULT__SPARE_0_SD1_HI                              = 0x0EF4,
    PREV_SHADOW_RESULT__SPARE_0_SD1_LO                              = 0x0EF5,
    PREV_SHADOW_RESULT__SPARE_1_SD1                                 = 0x0EF6,
    PREV_SHADOW_RESULT__SPARE_1_SD1_HI                              = 0x0EF6,
    PREV_SHADOW_RESULT__SPARE_1_SD1_LO                              = 0x0EF7,
    PREV_SHADOW_RESULT__SPARE_2_SD1                                 = 0x0EF8,
    PREV_SHADOW_RESULT__SPARE_2_SD1_HI                              = 0x0EF8,
    PREV_SHADOW_RESULT__SPARE_2_SD1_LO                              = 0x0EF9,
    PREV_SHADOW_RESULT__SPARE_3_SD1                                 = 0x0EFA,
    PREV_SHADOW_RESULT__SPARE_3_SD1_HI                              = 0x0EFA,
    PREV_SHADOW_RESULT__SPARE_3_SD1_LO                              = 0x0EFB,
    PREV_SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0              = 0x0EFC,
    PREV_SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_3            = 0x0EFC,
    PREV_SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_2            = 0x0EFD,
    PREV_SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_1            = 0x0EFE,
    PREV_SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_0            = 0x0EFF,
    PREV_SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD0               = 0x0F00,
    PREV_SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_3             = 0x0F00,
    PREV_SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_2             = 0x0F01,
    PREV_SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_1             = 0x0F02,
    PREV_SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_0             = 0x0F03,
    PREV_SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0                = 0x0F04,
    PREV_SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_3              = 0x0F04,
    PREV_SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_2              = 0x0F05,
    PREV_SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_1              = 0x0F06,
    PREV_SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_0              = 0x0F07,
    PREV_SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0              = 0x0F08,
    PREV_SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_3            = 0x0F08,
    PREV_SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_2            = 0x0F09,
    PREV_SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_1            = 0x0F0A,
    PREV_SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_0            = 0x0F0B,
    PREV_SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1              = 0x0F0C,
    PREV_SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_3            = 0x0F0C,
    PREV_SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_2            = 0x0F0D,
    PREV_SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_1            = 0x0F0E,
    PREV_SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_0            = 0x0F0F,
    PREV_SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD1               = 0x0F10,
    PREV_SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_3             = 0x0F10,
    PREV_SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_2             = 0x0F11,
    PREV_SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_1             = 0x0F12,
    PREV_SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_0             = 0x0F13,
    PREV_SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1                = 0x0F14,
    PREV_SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_3              = 0x0F14,
    PREV_SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_2              = 0x0F15,
```

```
    PREV_SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_1                  = 0x0F16,
    PREV_SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_0                  = 0x0F17,
    PREV_SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1                  = 0x0F18,
    PREV_SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_3                = 0x0F18,
    PREV_SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_2                = 0x0F19,
    PREV_SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_1                = 0x0F1A,
    PREV_SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_0                = 0x0F1B,
    PREV_SHADOW_RESULT_CORE__SPARE_0                                    = 0x0F1C,
    RESULT__DEBUG_STATUS                                                = 0x0F20,
    RESULT__DEBUG_STAGE                                                 = 0x0F21,
    GPH__SYSTEM__THRESH_RATE_HIGH                                       = 0x0F24,
    GPH__SYSTEM__THRESH_RATE_HIGH_HI                                    = 0x0F24,
    GPH__SYSTEM__THRESH_RATE_HIGH_LO                                    = 0x0F25,
    GPH__SYSTEM__THRESH_RATE_LOW                                        = 0x0F26,
    GPH__SYSTEM__THRESH_RATE_LOW_HI                                     = 0x0F26,
    GPH__SYSTEM__THRESH_RATE_LOW_LO                                     = 0x0F27,
    GPH__SYSTEM__INTERRUPT_CONFIG_GPIO                                  = 0x0F28,
    GPH__DSS_CONFIG__ROI_MODE_CONTROL                                   = 0x0F2F,
    GPH__DSS_CONFIG__MANUAL_EFFECTIVE_SPADS_SELECT                      = 0x0F30,
    GPH__DSS_CONFIG__MANUAL_EFFECTIVE_SPADS_SELECT_HI                   = 0x0F30,
    GPH__DSS_CONFIG__MANUAL_EFFECTIVE_SPADS_SELECT_LO                   = 0x0F31,
    GPH__DSS_CONFIG__MANUAL_BLOCK_SELECT                                = 0x0F32,
    GPH__DSS_CONFIG__MAX_SPADS_LIMIT                                    = 0x0F33,
    GPH__DSS_CONFIG__MIN_SPADS_LIMIT                                    = 0x0F34,
    GPH__MM_CONFIG__TIMEOUT_MACROP_A_HI                                 = 0x0F36,
    GPH__MM_CONFIG__TIMEOUT_MACROP_A_LO                                 = 0x0F37,
    GPH__MM_CONFIG__TIMEOUT_MACROP_B_HI                                 = 0x0F38,
    GPH__MM_CONFIG__TIMEOUT_MACROP_B_LO                                 = 0x0F39,
    GPH__RANGE_CONFIG__TIMEOUT_MACROP_A_HI                              = 0x0F3A,
    GPH__RANGE_CONFIG__TIMEOUT_MACROP_A_LO                              = 0x0F3B,
    GPH__RANGE_CONFIG__VCSEL_PERIOD_A                                   = 0x0F3C,
    GPH__RANGE_CONFIG__VCSEL_PERIOD_B                                   = 0x0F3D,
    GPH__RANGE_CONFIG__TIMEOUT_MACROP_B_HI                              = 0x0F3E,
    GPH__RANGE_CONFIG__TIMEOUT_MACROP_B_LO                              = 0x0F3F,
    GPH__RANGE_CONFIG__SIGMA_THRESH                                     = 0x0F40,
    GPH__RANGE_CONFIG__SIGMA_THRESH_HI                                  = 0x0F40,
    GPH__RANGE_CONFIG__SIGMA_THRESH_LO                                  = 0x0F41,
    GPH__RANGE_CONFIG__MIN_COUNT_RATE_RTN_LIMIT_MCPS                    = 0x0F42,
    GPH__RANGE_CONFIG__MIN_COUNT_RATE_RTN_LIMIT_MCPS_HI                 = 0x0F42,
    GPH__RANGE_CONFIG__MIN_COUNT_RATE_RTN_LIMIT_MCPS_LO                 = 0x0F43,
    GPH__RANGE_CONFIG__VALID_PHASE_LOW                                  = 0x0F44,
    GPH__RANGE_CONFIG__VALID_PHASE_HIGH                                 = 0x0F45,
    FIRMWARE__INTERNAL_STREAM_COUNT_DIV                                 = 0x0F46,
    FIRMWARE__INTERNAL_STREAM_COUNTER_VAL                               = 0x0F47,
    DSS_CALC__ROI_CTRL                                                  = 0x0F54,
    DSS_CALC__SPARE_1                                                   = 0x0F55,
    DSS_CALC__SPARE_2                                                   = 0x0F56,
    DSS_CALC__SPARE_3                                                   = 0x0F57,
    DSS_CALC__SPARE_4                                                   = 0x0F58,
    DSS_CALC__SPARE_5                                                   = 0x0F59,
```

```
    DSS_CALC__SPARE_6                                                = 0x0F5A,
    DSS_CALC__SPARE_7                                                = 0x0F5B,
    DSS_CALC__USER_ROI_SPAD_EN_0                                     = 0x0F5C,
    DSS_CALC__USER_ROI_SPAD_EN_1                                     = 0x0F5D,
    DSS_CALC__USER_ROI_SPAD_EN_2                                     = 0x0F5E,
    DSS_CALC__USER_ROI_SPAD_EN_3                                     = 0x0F5F,
    DSS_CALC__USER_ROI_SPAD_EN_4                                     = 0x0F60,
    DSS_CALC__USER_ROI_SPAD_EN_5                                     = 0x0F61,
    DSS_CALC__USER_ROI_SPAD_EN_6                                     = 0x0F62,
    DSS_CALC__USER_ROI_SPAD_EN_7                                     = 0x0F63,
    DSS_CALC__USER_ROI_SPAD_EN_8                                     = 0x0F64,
    DSS_CALC__USER_ROI_SPAD_EN_9                                     = 0x0F65,
    DSS_CALC__USER_ROI_SPAD_EN_10                                    = 0x0F66,
    DSS_CALC__USER_ROI_SPAD_EN_11                                    = 0x0F67,
    DSS_CALC__USER_ROI_SPAD_EN_12                                    = 0x0F68,
    DSS_CALC__USER_ROI_SPAD_EN_13                                    = 0x0F69,
    DSS_CALC__USER_ROI_SPAD_EN_14                                    = 0x0F6A,
    DSS_CALC__USER_ROI_SPAD_EN_15                                    = 0x0F6B,
    DSS_CALC__USER_ROI_SPAD_EN_16                                    = 0x0F6C,
    DSS_CALC__USER_ROI_SPAD_EN_17                                    = 0x0F6D,
    DSS_CALC__USER_ROI_SPAD_EN_18                                    = 0x0F6E,
    DSS_CALC__USER_ROI_SPAD_EN_19                                    = 0x0F6F,
    DSS_CALC__USER_ROI_SPAD_EN_20                                    = 0x0F70,
    DSS_CALC__USER_ROI_SPAD_EN_21                                    = 0x0F71,
    DSS_CALC__USER_ROI_SPAD_EN_22                                    = 0x0F72,
    DSS_CALC__USER_ROI_SPAD_EN_23                                    = 0x0F73,
    DSS_CALC__USER_ROI_SPAD_EN_24                                    = 0x0F74,
    DSS_CALC__USER_ROI_SPAD_EN_25                                    = 0x0F75,
    DSS_CALC__USER_ROI_SPAD_EN_26                                    = 0x0F76,
    DSS_CALC__USER_ROI_SPAD_EN_27                                    = 0x0F77,
    DSS_CALC__USER_ROI_SPAD_EN_28                                    = 0x0F78,
    DSS_CALC__USER_ROI_SPAD_EN_29                                    = 0x0F79,
    DSS_CALC__USER_ROI_SPAD_EN_30                                    = 0x0F7A,
    DSS_CALC__USER_ROI_SPAD_EN_31                                    = 0x0F7B,
    DSS_CALC__USER_ROI_0                                             = 0x0F7C,
    DSS_CALC__USER_ROI_1                                             = 0x0F7D,
    DSS_CALC__MODE_ROI_0                                             = 0x0F7E,
    DSS_CALC__MODE_ROI_1                                             = 0x0F7F,
    SIGMA_ESTIMATOR_CALC__SPARE_0                                    = 0x0F80,
    VHV_RESULT__PEAK_SIGNAL_RATE_MCPS                                = 0x0F82,
    VHV_RESULT__PEAK_SIGNAL_RATE_MCPS_HI                             = 0x0F82,
    VHV_RESULT__PEAK_SIGNAL_RATE_MCPS_LO                             = 0x0F83,
    VHV_RESULT__SIGNAL_TOTAL_EVENTS_REF                              = 0x0F84,
    VHV_RESULT__SIGNAL_TOTAL_EVENTS_REF_3                            = 0x0F84,
    VHV_RESULT__SIGNAL_TOTAL_EVENTS_REF_2                            = 0x0F85,
    VHV_RESULT__SIGNAL_TOTAL_EVENTS_REF_1                            = 0x0F86,
    VHV_RESULT__SIGNAL_TOTAL_EVENTS_REF_0                            = 0x0F87,
    PHASECAL_RESULT__PHASE_OUTPUT_REF                                = 0x0F88,
    PHASECAL_RESULT__PHASE_OUTPUT_REF_HI                             = 0x0F88,
    PHASECAL_RESULT__PHASE_OUTPUT_REF_LO                             = 0x0F89,
```

```
    DSS_RESULT__TOTAL_RATE_PER_SPAD                          = 0x0F8A,
    DSS_RESULT__TOTAL_RATE_PER_SPAD_HI                       = 0x0F8A,
    DSS_RESULT__TOTAL_RATE_PER_SPAD_LO                       = 0x0F8B,
    DSS_RESULT__ENABLED_BLOCKS                               = 0x0F8C,
    DSS_RESULT__NUM_REQUESTED_SPADS                          = 0x0F8E,
    DSS_RESULT__NUM_REQUESTED_SPADS_HI                       = 0x0F8E,
    DSS_RESULT__NUM_REQUESTED_SPADS_LO                       = 0x0F8F,
    MM_RESULT__INNER_INTERSECTION_RATE                       = 0x0F92,
    MM_RESULT__INNER_INTERSECTION_RATE_HI                    = 0x0F92,
    MM_RESULT__INNER_INTERSECTION_RATE_LO                    = 0x0F93,
    MM_RESULT__OUTER_COMPLEMENT_RATE                         = 0x0F94,
    MM_RESULT__OUTER_COMPLEMENT_RATE_HI                      = 0x0F94,
    MM_RESULT__OUTER_COMPLEMENT_RATE_LO                      = 0x0F95,
    MM_RESULT__TOTAL_OFFSET                                  = 0x0F96,
    MM_RESULT__TOTAL_OFFSET_HI                               = 0x0F96,
    MM_RESULT__TOTAL_OFFSET_LO                               = 0x0F97,
    XTALK_CALC__XTALK_FOR_ENABLED_SPADS                      = 0x0F98,
    XTALK_CALC__XTALK_FOR_ENABLED_SPADS_3                    = 0x0F98,
    XTALK_CALC__XTALK_FOR_ENABLED_SPADS_2                    = 0x0F99,
    XTALK_CALC__XTALK_FOR_ENABLED_SPADS_1                    = 0x0F9A,
    XTALK_CALC__XTALK_FOR_ENABLED_SPADS_0                    = 0x0F9B,
    XTALK_RESULT__AVG_XTALK_USER_ROI_KCPS                    = 0x0F9C,
    XTALK_RESULT__AVG_XTALK_USER_ROI_KCPS_3                  = 0x0F9C,
    XTALK_RESULT__AVG_XTALK_USER_ROI_KCPS_2                  = 0x0F9D,
    XTALK_RESULT__AVG_XTALK_USER_ROI_KCPS_1                  = 0x0F9E,
    XTALK_RESULT__AVG_XTALK_USER_ROI_KCPS_0                  = 0x0F9F,
    XTALK_RESULT__AVG_XTALK_MM_INNER_ROI_KCPS                = 0x0FA0,
    XTALK_RESULT__AVG_XTALK_MM_INNER_ROI_KCPS_3              = 0x0FA0,
    XTALK_RESULT__AVG_XTALK_MM_INNER_ROI_KCPS_2              = 0x0FA1,
    XTALK_RESULT__AVG_XTALK_MM_INNER_ROI_KCPS_1              = 0x0FA2,
    XTALK_RESULT__AVG_XTALK_MM_INNER_ROI_KCPS_0              = 0x0FA3,
    XTALK_RESULT__AVG_XTALK_MM_OUTER_ROI_KCPS                = 0x0FA4,
    XTALK_RESULT__AVG_XTALK_MM_OUTER_ROI_KCPS_3              = 0x0FA4,
    XTALK_RESULT__AVG_XTALK_MM_OUTER_ROI_KCPS_2              = 0x0FA5,
    XTALK_RESULT__AVG_XTALK_MM_OUTER_ROI_KCPS_1              = 0x0FA6,
    XTALK_RESULT__AVG_XTALK_MM_OUTER_ROI_KCPS_0              = 0x0FA7,
    RANGE_RESULT__ACCUM_PHASE                                = 0x0FA8,
    RANGE_RESULT__ACCUM_PHASE_3                              = 0x0FA8,
    RANGE_RESULT__ACCUM_PHASE_2                              = 0x0FA9,
    RANGE_RESULT__ACCUM_PHASE_1                              = 0x0FAA,
    RANGE_RESULT__ACCUM_PHASE_0                              = 0x0FAB,
    RANGE_RESULT__OFFSET_CORRECTED_RANGE                     = 0x0FAC,
    RANGE_RESULT__OFFSET_CORRECTED_RANGE_HI                  = 0x0FAC,
    RANGE_RESULT__OFFSET_CORRECTED_RANGE_LO                  = 0x0FAD,
    SHADOW_PHASECAL_RESULT__VCSEL_START                      = 0x0FAE,
    SHADOW_RESULT__INTERRUPT_STATUS                          = 0x0FB0,
    SHADOW_RESULT__RANGE_STATUS                              = 0x0FB1,
    SHADOW_RESULT__REPORT_STATUS                             = 0x0FB2,
    SHADOW_RESULT__STREAM_COUNT                              = 0x0FB3,
    SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD0            = 0x0FB4,
```

```
    SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD0_HI                          = 0x0FB4,
    SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD0_LO                          = 0x0FB5,
    SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD0                            = 0x0FB6,
    SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD0_HI                         = 0x0FB6,
    SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD0_LO                         = 0x0FB7,
    SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD0                                = 0x0FB8,
    SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD0_HI                             = 0x0FB8,
    SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD0_LO                             = 0x0FB9,
    SHADOW_RESULT__SIGMA_SD0                                                  = 0x0FBA,
    SHADOW_RESULT__SIGMA_SD0_HI                                               = 0x0FBA,
    SHADOW_RESULT__SIGMA_SD0_LO                                               = 0x0FBB,
    SHADOW_RESULT__PHASE_SD0                                                  = 0x0FBC,
    SHADOW_RESULT__PHASE_SD0_HI                                               = 0x0FBC,
    SHADOW_RESULT__PHASE_SD0_LO                                               = 0x0FBD,
    SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD0                     = 0x0FBE,
    SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD0_HI                  = 0x0FBE,
    SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD0_LO                  = 0x0FBF,
    SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_CROSSTALK_CORRECTED_MCPS_SD0        = 0x0FC0,
    SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_CROSSTALK_CORRECTED_MCPS_SD0_HI     = 0x0FC0,
    SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_CROSSTALK_CORRECTED_MCPS_SD0_LO     = 0x0FC1,
    SHADOW_RESULT__MM_INNER_ACTUAL_EFFECTIVE_SPADS_SD0                        = 0x0FC2,
    SHADOW_RESULT__MM_INNER_ACTUAL_EFFECTIVE_SPADS_SD0_HI                     = 0x0FC2,
    SHADOW_RESULT__MM_INNER_ACTUAL_EFFECTIVE_SPADS_SD0_LO                     = 0x0FC3,
    SHADOW_RESULT__MM_OUTER_ACTUAL_EFFECTIVE_SPADS_SD0                        = 0x0FC4,
    SHADOW_RESULT__MM_OUTER_ACTUAL_EFFECTIVE_SPADS_SD0_HI                     = 0x0FC4,
    SHADOW_RESULT__MM_OUTER_ACTUAL_EFFECTIVE_SPADS_SD0_LO                     = 0x0FC5,
    SHADOW_RESULT__AVG_SIGNAL_COUNT_RATE_MCPS_SD0                             = 0x0FC6,
    SHADOW_RESULT__AVG_SIGNAL_COUNT_RATE_MCPS_SD0_HI                          = 0x0FC6,
    SHADOW_RESULT__AVG_SIGNAL_COUNT_RATE_MCPS_SD0_LO                          = 0x0FC7,
    SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD1                             = 0x0FC8,
    SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD1_HI                          = 0x0FC8,
    SHADOW_RESULT__DSS_ACTUAL_EFFECTIVE_SPADS_SD1_LO                          = 0x0FC9,
    SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD1                            = 0x0FCA,
    SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD1_HI                         = 0x0FCA,
    SHADOW_RESULT__PEAK_SIGNAL_COUNT_RATE_MCPS_SD1_LO                         = 0x0FCB,
    SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD1                                = 0x0FCC,
    SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD1_HI                             = 0x0FCC,
    SHADOW_RESULT__AMBIENT_COUNT_RATE_MCPS_SD1_LO                             = 0x0FCD,
    SHADOW_RESULT__SIGMA_SD1                                                  = 0x0FCE,
    SHADOW_RESULT__SIGMA_SD1_HI                                               = 0x0FCE,
    SHADOW_RESULT__SIGMA_SD1_LO                                               = 0x0FCF,
    SHADOW_RESULT__PHASE_SD1                                                  = 0x0FD0,
    SHADOW_RESULT__PHASE_SD1_HI                                               = 0x0FD0,
    SHADOW_RESULT__PHASE_SD1_LO                                               = 0x0FD1,
    SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD1                     = 0x0FD2,
    SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD1_HI                  = 0x0FD2,
    SHADOW_RESULT__FINAL_CROSSTALK_CORRECTED_RANGE_MM_SD1_LO                  = 0x0FD3,
    SHADOW_RESULT__SPARE_0_SD1                                                = 0x0FD4,
    SHADOW_RESULT__SPARE_0_SD1_HI                                             = 0x0FD4,
    SHADOW_RESULT__SPARE_0_SD1_LO                                             = 0x0FD5,
```

```
      SHADOW_RESULT__SPARE_1_SD1                                    = 0x0FD6,
      SHADOW_RESULT__SPARE_1_SD1_HI                                 = 0x0FD6,
      SHADOW_RESULT__SPARE_1_SD1_LO                                 = 0x0FD7,
      SHADOW_RESULT__SPARE_2_SD1                                    = 0x0FD8,
      SHADOW_RESULT__SPARE_2_SD1_HI                                 = 0x0FD8,
      SHADOW_RESULT__SPARE_2_SD1_LO                                 = 0x0FD9,
      SHADOW_RESULT__SPARE_3_SD1                                    = 0x0FDA,
      SHADOW_RESULT__THRESH_INFO                                    = 0x0FDB,
      SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0                 = 0x0FDC,
      SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_3               = 0x0FDC,
      SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_2               = 0x0FDD,
      SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_1               = 0x0FDE,
      SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD0_0               = 0x0FDF,
      SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD0                  = 0x0FE0,
      SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_3                = 0x0FE0,
      SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_2                = 0x0FE1,
      SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_1                = 0x0FE2,
      SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD0_0                = 0x0FE3,
      SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0                   = 0x0FE4,
      SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_3                 = 0x0FE4,
      SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_2                 = 0x0FE5,
      SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_1                 = 0x0FE6,
      SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD0_0                 = 0x0FE7,
      SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0                 = 0x0FE8,
      SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_3               = 0x0FE8,
      SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_2               = 0x0FE9,
      SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_1               = 0x0FEA,
      SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD0_0               = 0x0FEB,
      SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1                 = 0x0FEC,
      SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_3               = 0x0FEC,
      SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_2               = 0x0FED,
      SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_1               = 0x0FEE,
      SHADOW_RESULT_CORE__AMBIENT_WINDOW_EVENTS_SD1_0               = 0x0FEF,
      SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD1                  = 0x0FF0,
      SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_3                = 0x0FF0,
      SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_2                = 0x0FF1,
      SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_1                = 0x0FF2,
      SHADOW_RESULT_CORE__RANGING_TOTAL_EVENTS_SD1_0                = 0x0FF3,
      SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1                   = 0x0FF4,
      SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_3                 = 0x0FF4,
      SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_2                 = 0x0FF5,
      SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_1                 = 0x0FF6,
      SHADOW_RESULT_CORE__SIGNAL_TOTAL_EVENTS_SD1_0                 = 0x0FF7,
      SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1                 = 0x0FF8,
      SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_3               = 0x0FF8,
      SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_2               = 0x0FF9,
      SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_1               = 0x0FFA,
      SHADOW_RESULT_CORE__TOTAL_PERIODS_ELAPSED_SD1_0               = 0x0FFB,
      SHADOW_RESULT_CORE__SPARE_0                                   = 0x0FFC,
      SHADOW_PHASECAL_RESULT__REFERENCE_PHASE_HI                    = 0x0FFE,
```

```
    SHADOW_PHASECAL_RESULT__REFERENCE_PHASE_LO                              = 0x0FFF,
  };

  enum DistanceMode { Short, Medium, Long, Unknown };

  enum RangeStatus : uint8_t
  {
    RangeValid                =   0,

    // "sigma estimator check is above the internal defined threshold"
    // (sigma = standard deviation of measurement)
    SigmaFail                 =   1,

    // "signal value is below the internal defined threshold"
    SignalFail                =   2,

    // "Target is below minimum detection threshold."
    RangeValidMinRangeClipped =   3,

    // "phase is out of bounds"
    // (nothing detected in range; try a longer distance mode if applicable)
    OutOfBoundsFail           =   4,

    // "HW or VCSEL failure"
    HardwareFail              =   5,

    // "The Range is valid but the wraparound check has not been done."
    RangeValidNoWrapCheckFail =   6,

    // "Wrapped target, not matching phases"
    // "no matching phase in other VCSEL period timing."
    WrapTargetFail            =   7,

    // "Internal algo underflow or overflow in lite ranging."
  // ProcessingFail            =   8: not used in API

    // "Specific to lite ranging."
    // should never occur with this lib (which uses low power auto ranging,
    // as the API does)
    XtalkSignalFail           =   9,

    // "1st interrupt when starting ranging in back to back mode. Ignore
    // data."
    // should never occur with this lib
    SynchronizationInt         =  10, // (the API spells this "syncronisation")

    // "All Range ok but object is result of multiple pulses merging together.
    // Used by RQL for merged pulse detection"
  // RangeValid MergedPulse    =  11: not used in API
```

```
      // "Used by RQL as different to phase fail."
   // TargetPresentLackOfSignal =  12:

      // "Target is below minimum detection threshold."
      MinRangeFail              =  13,

      // "The reported range is invalid"
   // RangeInvalid              =  14: can't actually be returned by API (range can never become negative, even after
correction)

      // "No Update."
      None                      = 255,
    };

    struct RangingData
    {
      uint16_t range_mm;
      RangeStatus range_status;
      float peak_signal_count_rate_MCPS;
      float ambient_count_rate_MCPS;
    };

    RangingData ranging_data;

    uint8_t last_status; // status of last I2C transmission

    VL53L1X();

    void setAddress(uint8_t new_addr);
    uint8_t getAddress() { return address; }

    bool init(bool io_2v8 = true);

    void writeReg(uint16_t reg, uint8_t value);
    void writeReg16Bit(uint16_t reg, uint16_t value);
    void writeReg32Bit(uint16_t reg, uint32_t value);
    uint8_t readReg(regAddr reg);
    uint16_t readReg16Bit(uint16_t reg);
    uint32_t readReg32Bit(uint16_t reg);

    bool setDistanceMode(DistanceMode mode);
    DistanceMode getDistanceMode() { return distance_mode; }

    bool setMeasurementTimingBudget(uint32_t budget_us);
    uint32_t getMeasurementTimingBudget();

    void startContinuous(uint32_t period_ms);
    void stopContinuous();
    uint16_t read(bool blocking = true);
    uint16_t readRangeContinuousMillimeters(bool blocking = true) { return read(blocking); } // alias of read()
```

```
    // check if sensor has new reading available
    // assumes interrupt is active low (GPIO_HV_MUX__CTRL bit 4 is 1)
    bool dataReady() { return (readReg(GPIO__TIO_HV_STATUS) & 0x01) == 0; }

    static const char * rangeStatusToString(RangeStatus status);

    void setTimeout(uint16_t timeout) { io_timeout = timeout; }
    uint16_t getTimeout() { return io_timeout; }
    bool timeoutOccurred();

  private:

    // The Arduino two-wire interface uses a 7-bit number for the address,
    // and sets the last bit correctly based on reads and writes
    static const uint8_t AddressDefault = 0b0101001;

    // value used in measurement timing budget calculations
    // assumes PresetMode is LOWPOWER_AUTONOMOUS
    //
    // vhv = LOWPOWER_AUTO_VHV_LOOP_DURATION_US + LOWPOWERAUTO_VHV_LOOP_BOUND
    //       (tuning parm default) * LOWPOWER_AUTO_VHV_LOOP_DURATION_US
    //     = 245 + 3 * 245 = 980
    // TimingGuard = LOWPOWER_AUTO_OVERHEAD_BEFORE_A_RANGING +
    //               LOWPOWER_AUTO_OVERHEAD_BETWEEN_A_B_RANGING + vhv
    //             = 1448 + 2100 + 980 = 4528
    static const uint32_t TimingGuard = 4528;

    // value in DSS_CONFIG__TARGET_TOTAL_RATE_MCPS register, used in DSS
    // calculations
    static const uint16_t TargetRate = 0x0A00;

    // for storing values read from RESULT__RANGE_STATUS (0x0089)
    // through RESULT__PEAK_SIGNAL_COUNT_RATE_CROSSTALK_CORRECTED_MCPS_SD0_LOW
    // (0x0099)
    struct ResultBuffer
    {
      uint8_t range_status;
    // uint8_t report_status: not used
      uint8_t stream_count;
      uint16_t dss_actual_effective_spads_sd0;
   // uint16_t peak_signal_count_rate_mcps_sd0: not used
      uint16_t ambient_count_rate_mcps_sd0;
   // uint16_t sigma_sd0: not used
   // uint16_t phase_sd0: not used
      uint16_t final_crosstalk_corrected_range_mm_sd0;
      uint16_t peak_signal_count_rate_crosstalk_corrected_mcps_sd0;
    };

    // making this static would save RAM for multiple instances as long as there
```

```
    // aren't multiple sensors being read at the same time (e.g. on separate
    // I2C buses)
    ResultBuffer results;

    uint8_t address;

    uint16_t io_timeout;
    bool did_timeout;
    uint16_t timeout_start_ms;

    uint16_t fast_osc_frequency;
    uint16_t osc_calibrate_val;

    bool calibrated;
    uint8_t saved_vhv_init;
    uint8_t saved_vhv_timeout;

    DistanceMode distance_mode;

    // Record the current time to check an upcoming timeout against
    void startTimeout() { timeout_start_ms = millis(); }

    // Check if timeout is enabled (set to nonzero value) and has expired
    bool checkTimeoutExpired() {return (io_timeout > 0) && ((uint16_t)(millis() - timeout_start_ms) > io_timeout); }

    void setupManualCalibration();
    void readResults();
    void updateDSS();
    void getRangingData();

    static uint32_t decodeTimeout(uint16_t reg_val);
    static uint16_t encodeTimeout(uint32_t timeout_mclks);
    static uint32_t timeoutMclksToMicroseconds(uint32_t timeout_mclks, uint32_t macro_period_us);
    static uint32_t timeoutMicrosecondsToMclks(uint32_t timeout_us, uint32_t macro_period_us);
    uint32_t calcMacroPeriod(uint8_t vcsel_period);

    // Convert count rate from fixed point 9.7 format to float
    float countRateFixedToFloat(uint16_t count_rate_fixed) { return (float)count_rate_fixed / (1 << 7); }
};
```

```
/*
This example shows how to take simple range measurements with the VL53L1X. The
range readings are in units of mm.
*/

#include <Wire.h>
#include <VL53L1X.h>

VL53L1X sensor;

void setup()
{
  Serial.begin(115200);
  Wire.begin();
  Wire.setClock(400000); // use 400 kHz I2C

  sensor.setTimeout(500);
  if (!sensor.init())
  {
    Serial.println("Failed to detect and initialize sensor!");
    while (1);
  }

  // Use long distance mode and allow up to 50000 us (50 ms) for a measurement.
  // You can change these settings to adjust the performance of the sensor, but
  // the minimum timing budget is 20 ms for short distance mode and 33 ms for
  // medium and long distance modes. See the VL53L1X datasheet for more
  // information on range and timing limits.
  sensor.setDistanceMode(VL53L1X::Long);
  sensor.setMeasurementTimingBudget(50000);

  // Start continuous readings at a rate of one measurement every 50 ms (the
  // inter-measurement period). This period should be at least as long as the
  // timing budget.
  sensor.startContinuous(50);
}

void loop()
{
  Serial.print(sensor.read());
  if (sensor.timeoutOccurred()) { Serial.print(" TIMEOUT"); }

  Serial.println();
}
```