1. ***Key APIs Notes***

void IIM4623x\_set\_serif

***Function***:

*Set serial interface to send command to device and receive response from device.*

***Parameters***:

uint32\_t (\*rx)(uint8\_t \*, uint32\_t),

*The function pointer of Rx API depending on platform system via SPI or UART.*

uint32\_t (\*tx)(uint8\_t \*, uint32\_t),

*The function pointer of Tx API depending on platform system via SPI or UART.*

uint32\_t (\*tx\_rx)(uint8\_t \*, uint32\_t, uint8\_t \*, uint32\_t)

*The function pointer of Tx/Rx API depending on platform system via SPI or UART.*

***Return***:

*NULL*.

void IIM4623x\_set\_reset

***Function***:

*Set serial interface to reset device.*

***Parameters***:

void (\*reset)(void)

*The function pointer of Reset API depending on platform system.*

***Return***:

*NULL*.

uint32\_t IIM4623x\_ serif\_rx

***Function***:

*Wrapper function of Rx API to receive response from device in command mode. Half-duplex communication is used.*

***Parameters***:

uint8\_t \*buf,

*Pointer to packet buffer for Rx.*

uint32\_t len

*Size of buffer.*

***Return***:

*0: Success, Others: Fail*.

uint32\_t IIM4623x\_ serif\_tx

***Function***:

*Wrapper function of Tx API to send command to device in command mode. Half-duplex communication is used.*

***Parameters***:

uint8\_t \*buf,

*Pointer to packet buffer for Tx.*

uint32\_t len

*Size of buffer.*

***Return***:

*0: Success, Others: Fail*.

uint32\_t IIM4623x\_ serif\_tx\_rx

***Function***:

*Wrapper function of Tx/Rx API to send command to device and receive response from device in streaming mode. Full-duplex communication is used for transmitting and receiving IMU data.*

***Parameters***:

uint8\_t \*wbuf,

*Pointer to packet buffer for Tx.*

uint32\_t wlen,

*Size of Tx buffer.*

uint8\_t \*rbuf,

*Pointer to packet buffer for Rx.*

uint32\_t rlen

*Size of Rx buffer.*

***Return***:

*0: Success, Others: Fail*.

void IIM4623x\_Reset

***Function***:

*Wrapper function of Reset API.*

***Parameters***:

*NULL.*

***Return***:

*NULL*.

PartNum IIM4623x\_Get\_PartNum

***Function***:

*Return part number.*

***Parameters***:

*NULL.*

***Return***:

*0: IIM46230, 1: IIM46234*.

void IIM4623x\_SetCMD\_Firmware

***Function***:

*Set command packet for firmware update.*

***Parameters***:

uint8\_t cmd\_type

*Type of command.*

***Return***:

*NULL*.

void IIM4623x\_Switch\_to\_Bootloader

***Function***:

*Set UPGRADE\_REQUEST flag in the flash.*

***Parameters***:

*NULL*.

***Return***:

*NULL*.

void IIM4623x\_Upgrade\_Firmware

***Function***:

*Bootloader prepares Firmware upgrade.*

***Parameters***:

*NULL*.

***Return***:

*NULL*.

void IIM4623x\_Send\_Length\_Info

***Function***:

*Send information on length of firmware image to bootloader.*

***Parameters***:

*NULL*.

***Return***:

*NULL*.

void IIM4623x\_Send\_Image\_Data

***Function***:

*Send packet containing firmware image data to bootloader.*

***Parameters***:

*NULL*.

***Return***:

*NULL*.

void IIM4623x\_Clear\_Upgrade\_Flag

***Function***:

*Clear UPGRADE\_REQUEST flag in the flash.*

***Parameters***:

*NULL*.

***Return***:

*NULL*.

void IIM4623x\_Initialize

***Function***:

*Initialize variables for driver status. Reset device. Check WHO\_AM\_I.*

***Parameters***:

*NULL.*

***Return***:

*NULL*.

void IIM4623x\_SetCMD\_Common

***Function***:

*Set packet for common command.*

***Parameters***:

uint8\_t cmd\_type

*Type of command.*

***Return***:

*NULL*.

void IIM4623x\_SetCMD\_ReadRegister

***Function***:

*Set command packet to read register.*

***Parameters***:

reg user\_reg

*User register.*

***Return***:

*NULL*.

void IIM4623x\_ SetCMD\_WriteRegister

***Function***:

*Set command packet to write register.*

***Parameters***:

reg user\_reg,

*User register.*

uint8\_t \*value

*Pointer to value to be written to register.*

***Return***:

*NULL*.

void IIM4623x\_Set\_UtcTime

***Function***:

*Set command packet to set UTC time.*

***Parameters***:

utc time

*UTC time to be set.*

***Return***:

*NULL*.

void IIM4623x\_Read\_WhoAmI

***Function***:

*Set and send command packet to read WHO\_AM\_I from register.*

***Parameters***:

*NULL.*

***Return***:

*NULL*.

void IIM4623x\_Get\_SerialNum

***Function***:

*Set and send command packet to read serial number from register.*

***Parameters***:

*NULL.*

***Return***:

*NULL*.

void IIM4623x\_Get\_Version

***Function***:

*Set and send command packet to read firmware version from register.*

***Parameters***:

*NULL.*

***Return***:

*NULL*.

void IIM4623x\_Imu\_SelfTest

***Function***:

*Set and send command packet to invoke self-test.*

***Parameters***:

*NULL.*

***Return***:

*NULL*.

void IIM4623x\_Set\_OutDataForm

***Function***:

*Set and send command packet to set output data form.*

***Parameters***:

enum IIM4623x\_OutDataForm form

*Output data form to be set.*

*(0: 32-Bit IEEE 754 single-precision floating point,*

*1: 32-Bit Fixed point 2's Complement representation).*

***Return***:

*NULL*.

void IIM4623x\_Set\_SampleRateDiv

***Function***:

*Set and send command packet to set sample rate divisor.*

***Parameters***:

enum IIM4623x\_SampleRateDiv divisor

*Sample rate divisor for each ODR (1 ~ 100).*

***Return***:

*NULL*.

void IIM4623x\_Toggle\_DataOutput

***Function***:

*Set and send command packet to toggle data output.*

***Parameters***:

enum IIM4623x\_DataOutPut output

*0: Accel, 1: Gyro, 2: Temperature, 3: Delta Velocity, 4: Delta Angle.*

***Return***:

*NULL*.

void IIM4623x\_Set\_UartIfConfig

***Function***:

*Set and send command packet to set UART baud rate.*

***Parameters***:

enum IIM4623x\_UartBaudRate baud

*0: 921600, 1: 1500000, 2: 2000000, 3: 3000000.*

***Return***:

*NULL*.

void IIM4623x\_Set\_SyncConfig

***Function***:

*Set and send command packet to enable/disable PPS interface.*

***Parameters***:

enum IIM4623x\_SyncConfig sync\_mode

*0: Disable PPS, 1: Enable PPS.*

***Return***:

*NULL*.

void IIM4623x\_Set\_BWConfig\_Accel

***Function***:

*Set and send command packet to set LPF BW of Accel.*

***Parameters***:

enum IIM4623x\_AccBwConfig bw

*0x40: BW4, 0x50: BW5, 0x60: BW6, 0x70: BW7.*

***Return***:

*NULL*.

void IIM4623x\_Set\_BWConfig\_Gyro

***Function***:

*Set and send command packet to set LPF BW of Gyro.*

***Parameters***:

enum IIM4623x\_GyroBwConfig bw

*0x4: BW4, 0x5: BW5, 0x6: BW6, 0x7: BW7.*

***Return***:

*NULL*.

void IIM4623x\_Set\_AccelConfig0

***Function***:

*Set and send command packet to set FSR of Accel.*

***Parameters***:

enum IIM4623x\_AccelConfig0 fsr

*0x00: 16G, 0x20: 8G, 0x40: 4G, 0x60: 2G.*

***Return***:

*NULL*.

void IIM4623x\_Set\_GyroConfig0

***Function***:

*Set and send command packet to set FSR of Gyro.*

***Parameters***:

enum IIM4623x\_GyroConfig0 fsr

*0x00: 2000 DPS, 0x20: 1000 DPS, 0x40: 500(480) DPS, 0x60: 2000 DPS.*

***Return***:

*NULL*.

void IIM4623x\_Toggle\_ExtCalibConfig

***Function***:

*Set and send command packet to enable/disable bias & sensitivity calibration.*

***Parameters***:

enum IIM4623x\_CalibConfig config

*Decide whether bias & sensitivity calibration for Accel and Gyro is enabled or not.*

***Return***:

*NULL*.

void IIM4623x\_Write\_Bias

***Function***:

*Set and send command packet to write bias to register.*

***Parameters***:

enum IIM4623x\_Axis axis,

*0: Accel X, 1: Accel Y, 2: Accel Z, 3: Gyro X, 4: Gyro Y, 5: Gyro Z.*

float bias\_val

*Bias value to be written.*

***Return***:

*NULL*.

void IIM4623x\_Read\_Bias

***Function***:

*Set and send command packet to read bias from register.*

***Parameters***:

enum IIM4623x\_Axis axis,

*0: Accel X, 1: Accel Y, 2: Accel Z, 3: Gyro X, 4: Gyro Y, 5: Gyro Z.*

***Return***:

*NULL*.

void IIM4623x\_Write\_Sens\_Mat\_Accel

***Function***:

*Set and send command packet to write sensitivity matrix to register.*

***Parameters***:

enum IIM4623x\_Mat\_Index indx

*Index of sensitivity matrix.*

float value

*Value to be written to sensitivity matrix register.*

***Return***:

*NULL*.

void IIM4623x\_Write\_Sens\_Mat\_Gyro

***Function***:

*Set and send command packet to write sensitivity matrix to register.*

***Parameters***:

enum IIM4623x\_Mat\_Index indx

*Index of sensitivity matrix.*

float value

*Value to be written to sensitivity matrix register.*

***Return***:

*NULL*.

void IIM4623x\_Read\_Sens\_Mat\_Accel

***Function***:

*Set and send command packet to read sensitivity matrix from register.*

***Parameters***:

enum IIM4623x\_Mat\_Index indx

*Index of sensitivity matrix.*

***Return***:

*NULL*.

void IIM4623x\_Read\_Sens\_Mat\_Gyro

***Function***:

*Set and send command packet to read sensitivity matrix from register.*

***Parameters***:

enum IIM4623x\_Mat\_Index indx

*Index of sensitivity matrix.*

***Return***:

*NULL*.

void IIM4623x\_Start\_Streaming

***Function***:

*Set and send command packet to start data streaming.*

***Parameters***:

*NULL*.

***Return***:

*NULL*.

void IIM4623x\_Stop\_Streaming

***Function***:

*Set and send command packet to stop data streaming.*

***Parameters***:

*NULL*.

***Return***:

*NULL*.

void IIM4623x\_Set\_SensorFT

***Function***:

*Set and send command packet to enable/disable the sensorFT™ feature.*

***Parameters***:

bool on\_off

*Decide whether the sensorFT™ feature is enabled or not.*

***Return***:

*NULL*.

Uint32\_t IIM4623x\_Read\_DataOutput

***Function***:

*Parse packet of data stream and read output data when data ready interrupt happens.*

***Parameters***:

*NULL*.

***Return***:

*0: Success. Others: Fail.*

bool Is\_Command\_Mode

***Function***:

*Check whether current mode is command mode or not.*

***Parameters***:

*NULL*.

***Return***:

*true: command mode. false: streaming mode.*

void IIM4623x\_Set\_StopStreaming

***Function***:

*Set status variable which indicates that ‘STOP STREAMING’ command will be sent.*

***Parameters***:

*NULL*.

***Return***:

*NULL*.

void IIM4623x\_Set\_StartStreaming

***Function***:

*Set status variable which indicates that ‘START STREAMING’ command will be sent.*

***Parameters***:

*NULL*.

***Return***:

*NULL*.

Uint32\_t response\_handler

***Function***:

*Handle all responses from device when data ready interrupt happens.*

***Parameters***:

*NULL*.

***Return***:

*0: Success. Others: Fail.*

1. ***External Global Variables***

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| volatile int | drdy\_interrupt | Indicate that data ready interrupt happened. |
| volatile uint64\_t | timestamp | Timestamp of host system |
| int | bus\_type | 0: SPI, 1: UART |

1. ***Sample Code for Customer***

|  |
| --- |
| ***Main function of the example application which receives input from the keyboard***  void char\_input\_handler(uint8\_t input)  {  if (Is\_Command\_Mode()) {  switch (input) {  case 'r':  IIM4623x\_Initialize();  printf("Device was reset \n\r");  break;  case 'w':  IIM4623x\_Read\_WhoAmI();  printf("Sent command to read WHO\_AM\_I \n\r");  break;  case 'v':  IIM4623x\_Get\_Version();  printf("Sent command to get firmware version \n\r");  break;  break;  case 'f':  printf("\r\n");  printf(" Press '0' to select Data Output Form as Floating IEEE 754 \r\n");  printf(" Press '1' to select Data Output Form as Fixed 2's Complement \r\n");  printf("\r\n");  case 'o':  printf("\r\n");  printf(" Press '0' to set ODR @ 10Hz \r\n");  printf(" Press '1' to set ODR @ 20Hz \r\n");  printf(" Press '2' to set ODR @ 25Hz \r\n");  printf(" Press '3' to set ODR @ 50Hz \r\n");  printf(" Press '4' to set ODR @ 100Hz \r\n");  printf(" Press '5' to set ODR @ 125Hz \r\n");  printf(" Press '6' to set ODR @ 200Hz \r\n");  printf(" Press '7' to set ODR @ 250Hz \r\n");  printf(" Press '8' to set ODR @ 500Hz \r\n");  printf(" Press '9' to set ODR @ 1kHz \r\n");  printf("\r\n");  break;  case 'd':  printf("\r\n");  printf(" Press '0' to toggle Accel \r\n");  printf(" Press '1' to toggle Gyro \r\n");  printf(" Press '2' to toggle Temperature \r\n");  printf(" Press '3' to toggle Delta Velocity \r\n");  printf(" Press '4' to toggle Delta Angle \r\n");  printf("\r\n");  break;  case 's':  IIM4623x\_Set\_StartStreaming();  IIM4623x\_Start\_Streaming();  break;  default:  break;  }  }  else {  switch (input) {  case 'p':  IIM4623x\_Set\_StopStreaming();  break;  default:  break;  }  }  }  int main(void)  {  uint8\_t uc\_key = 0;  uint8\_t uc\_pre\_key = 0;  drdy\_interrupt = 0;  /\* Set R/W functions provided by serial interfaces (SPI/UART) of platform \*/  if (bus\_type == 0) {  IIM4623x\_set\_serif(spi\_master\_read, spi\_master\_write, spi\_master\_write\_read);  }  else if (bus\_type == 1) {  IIM4623x\_set\_serif(uart\_receive, uart\_transmit, uart\_rx\_tx);  }  /\* Set reset function which toggles GPIO for RESET pin \*/  IIM4623x\_set\_reset(device\_reset);  IIM4623x\_Initialize();  while (1) {  /\* Set ‘uc\_key’ with key input here \*/  switch(uc\_pre\_key) {  case 'f':  select\_data\_output\_form(uc\_key);  break;  case 'o':  select\_odr(uc\_key);  break;  case 'd':  select\_data\_output(uc\_key);  break;  default:  char\_input\_handler(uc\_key);  break;  }    uc\_pre\_key = uc\_key;  if (drdy\_interrupt == 1) {  drdy\_interrupt = 0;  if (response\_handler()) {  break;  }  }  }  } |

|  |
| --- |
| ***Select form of data output***  void select\_data\_output\_form(uint8\_t input)  {  if (Is\_Command\_Mode()) {  switch (input) {  case '0':  IIM4623x\_Set\_OutDataForm(FLOATING);  break;  case '1':  IIM4623x\_Set\_OutDataForm(FIXED);  break;  default:  break;  }  }  } |

|  |
| --- |
| ***Select ODR***  void select\_odr(uint8\_t input)  {  if (Is\_Command\_Mode()) {  switch (input) {  case '0':  IIM4623x\_Set\_SampleRateDiv(ODR\_10HZ);  break;  case '1':  IIM4623x\_Set\_SampleRateDiv(ODR\_20HZ);  break;  case '2':  IIM4623x\_Set\_SampleRateDiv(ODR\_25HZ);  break;  case '3':  IIM4623x\_Set\_SampleRateDiv(ODR\_50HZ);  break;  case '4':  IIM4623x\_Set\_SampleRateDiv(ODR\_100HZ);  break;  case '5':  IIM4623x\_Set\_SampleRateDiv(ODR\_125HZ);  break;  case '6':  IIM4623x\_Set\_SampleRateDiv(ODR\_200HZ);  break;  case '7':  IIM4623x\_Set\_SampleRateDiv(ODR\_250HZ);  break;  case '8':  IIM4623x\_Set\_SampleRateDiv(ODR\_500HZ);  break;  case '9':  IIM4623x\_Set\_SampleRateDiv(ODR\_1KHZ);  break;  default:  break;  }  }  } |

|  |
| --- |
| ***Select streaming data output***  void select\_data\_output(uint8\_t input)  {  if (Is\_Command\_Mode()) {  switch (input) {  case '0':  IIM4623x\_Toggle\_DataOutput(ACCEL);  break;  case '1':  IIM4623x\_Toggle\_DataOutput(GYRO);  break;  case '2':  IIM4623x\_Toggle\_DataOutput(TEMP);  break;  case '3':  IIM4623x\_Toggle\_DataOutput(DELTA\_VEL);  break;  case '4':  IIM4623x\_Toggle\_DataOutput(DELTA\_ANG);  break;  default:  break;  }  }  } |