

Ciontek

Development Specification for CM30

V1.0.7

Updated on
2025/01/09

| | |
|--|----|
| • 1. Introduction | 4 |
| • 2. Revisions | 7 |
| • 3. MDB Master | 7 |
| 3.1 Import the MDB Master SDK for Android Studio | 8 |
| 3.2. mdbMaster APIs | 8 |
| 3.2.1 Open | 8 |
| 3.2.2 Close | 9 |
| 3.2.3 sendAnswer | 9 |
| 3.2.4 sendCommand | 9 |
| 3.2.5 setLights | 10 |
| 3.3 Executive actuator | 10 |
| 3.3.1 gpio_set_digout_value | 10 |
| 3.3.2 gpio_set_wakeup_value | 10 |
| 3.3.3 gpio_get_digin_value | 11 |
| 3.4 Return code | 11 |
| • 4. MDB Slave | 11 |
| 4.1. MDB Slave Transport Layer APIs | 12 |
| 4.1.1 open | 12 |
| 4.1.2 close | 12 |
| 4.1.3 receiveCommand | 13 |
| 4.1.4 sendAnswer | 13 |
| 4.1.5 sendResponseData | 13 |
| 4.2. MDB Cashless Device APIs | 14 |
| • 5. RS232 | 14 |
| 5.1 RS232Open | 14 |
| 5.2 RS232Close | 15 |
| 5.3 RS232Write | 15 |
| 5.4 RS232Read | 15 |
| • 6. Digital IO | 16 |
| 6.1 digital_out_set_value | 16 |
| 6.2 digital_out_pulse | 16 |
| 6.3 digital_in_get_value | 17 |
| 6.4 digital_in_request_irq | 17 |

| | |
|---|----|
| 6.5 registerDigitalInterruptMonitor | 17 |
| 6.6 unregisterDigitalInterruptMonitor | 18 |
| 6.7 Return code | 18 |
| • 7. Led Light | 19 |
| 7.1 getLightInstance | 19 |
| 7.2 setLights | 19 |
| 7.3 destroy | 19 |
| • 8. Physical Key | 20 |
| • 9. USB2COM | 20 |
| 9.1 Open | 21 |
| 9.2 Close | 21 |
| 9.3 Write | 21 |
| 9.4 Read | 22 |

• 1. Introduction

Hereafter is some general instruction for your app development of CM30. The rest common features of SDK such as IC card reader please refer to the Ciontek general SDK instruction file, which you may access from your sales connection.

1. How to debug via USB ?

There is one Type-A port and one Type-C port working on Host and Device mode respectively. By Default the CM30 works in host mode with USB-A activated. The USB-C can't be used simultaneously with USB-A, and the following steps will help developer to debug terminal by device mode via USB-C port:

1. Enter 'Settings-Connected devices', choose "Device" in Usb mode
2. 'Settings-About device', quick click Build number and open "developer option"
3. 'Settings-Developer options', open USB debugging
4. Connect to PC via USB cable

2. How to debug via Wireless ?

If developer needs to develop application by USB Host mode, Wireless Adb debugging is recommended:

1. Switch the USB mode to device mode and open USB debugging according to the previous steps.
2. Connect to PC via USB cable
3. Input 'adb tcpip 5555'
4. Input 'adb connect 192.168.1.XXX:5555'
5. You can take off USB cable after it appeared "adb connect 192.168.1.XXX:5555"
6. Switch back to USB Host mode and start ADB debug

3. About MDB

CM30 supports one MDB Master port and one MDB Slave port

When CM30 is used as Vending Machine Controller (VMC), it is through MDB Master port to connect with peripheral equipment, please follow chapter 3 to import the "CM30-HardwareLibrary-1.0.3.aar" to your project and develop the application.

When CM30 is used as MDB Slave device, it is through MDB Slave port to connect with VMC please follow chapter 4 to import the "CM30-HardwareLibrary-1.0.3.aar" to your project and develop the application.

Note: MDB Master and MDB Slave can't be used simultaneously because of hardware design.

4. About RS232

CM30 supports two RS232 ports, RS232-A and RS232-B, please refer to the chapter 5 to develop the application.

The RS232-A supports 12-48V power inputs to power the CM30.

The RS232-B outputs 12-48v to supply peripheral devices.

Note: RS232-A and RS232-B can't be used simultaneously because of hardware design.

5. About Ethernet

The RJ45 port only works in USB Host mode, not work in Device mode.

6. About Digital port

The Digital port provided One output pin and three input pins which allows the CM30 support some pulse protocol devices. (Refer to the 'TABLE : Digital IO' for detailed pin description)

7. About the SMA

If the CM30 is used in a closed cabinet that may cause the GPS, 4G, wifi signal to be weakened, CM30 provides three SMA ports for you to connect to the external antenna.

Note: Before connecting an External Antenna via SMA, you need to select "External Antenna" in "Settings" -> "Network&internet" -> "Antenna switcher"

8. About IC Card

If you need to develop IC card, please contact sales to provide relevant development documents and SDK

9. About Pin Description

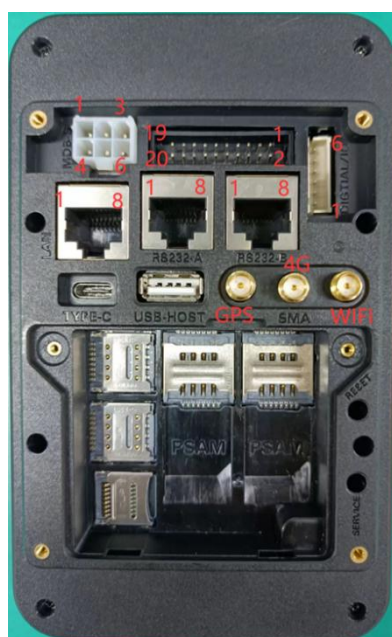


TABLE : LAN(Ethernet)

| Pin No. | Signal | Description |
|---------|--------|------------------------|
| 1 | TX+ | Transmit Data Positive |
| 2 | TX- | Transmit Data Negative |
| 3 | RX+ | Receive Data Positive |
| 4 | NC | |
| 5 | NC | |
| 6 | RX- | Receive Data Negative |
| 7 | NC | |
| 8 | NC | |

TABLE : RS232-A

| Pin No. | Signal | Description |
|---------|--------|-----------------------------|
| 1 | DC-IN | Power Supply (12V-48V/ 2A) |
| 2 | RX | Receive Data |
| 3 | TX | Transmit Data |
| 4 | NC | |
| 5 | RTS | Request To Send |
| 6 | CTS | Clear To Send |
| 7 | GND | Ground |
| 8 | GND | Ground |

TABLE : RS232-B

| Pin No. | Signal | Description |
|---------|--------|--|
| 1 | DC-OUT | Outputs (12V-48V/1A) To Supply Peripheral Devices |
| 2 | RX | Receive Data |
| 3 | TX | Transmit Data |
| 4 | NC | |
| 5 | RTS | Request To Send |
| 6 | CTS | Clear To Send |
| 7 | GND | Ground |
| 8 | GND | Ground |

TABLE : MDB Master/Executive

***PIN 3 - VMOUT : Direct output 12-48V when CM30 Power On**

**** PIN 18 - VDCOUT: Output 12-48V when called Master API 'open' (See chapter 3.2.1)**

| Pin No. | Signal | Description |
|---------|---------------|---|
| 1 | MDB-VDCIN | Power Supply (12V-48V /2A) |
| 2 | GND | Ground |
| 3 | VMOUT | Reserved Outputs (12V-48V/1A) |
| 4 | GND | Ground |
| 5 | MDB-MASTER-RX | Receive Data For MDB Master |
| 6 | MDB-MASTER-TX | Transmit Data For MDB Master |
| 7 | GND | Ground |
| 8 | GND | Ground |
| 9 | EXE-MASTER-RX | Receive Data For Executive Actuator (Reserved) |
| 10 | EXE-MASTER-TX | Transmit Data For Executive Actuator (Reserved) |
| 11 | GND | Ground |
| 12 | GND | Ground |
| 13 | DIGI-IN4-H | Reserved For Executive Actuator |
| 14 | DIGI-OUT2-H | Reserved For Executive Actuator |
| 15 | DIGI-IN4-L | Reserved For Executive Actuator |
| 16 | DIGI-OUT2-L | Reserved For Executive Actuator |
| 17 | MDB-WAKEUP | Output IO (5V TTL) Reserved For MDB |
| 18 | VDCOUT | Outputs (12V-48V/1A) To Supply Slave Devices |

| | | |
|----|------------|---------------------------------|
| 19 | EXE-ACIN_L | Reserved For Executive Actuator |
| 20 | EXE-ACIN_N | Reserved For Executive Actuator |

TABLE : MDB Slave

| Pin No. | Signal | Description |
|---------|------------|--|
| 1 | VDC-MDB | Power Supply (12V-48V /2A) |
| 2 | GND | Ground |
| 3 | NC | |
| 4 | MASTER-RX | Master Receive (MDB Slave Transmit Data) |
| 5 | MASTER-TX | Master Transmit (MDB Slave Receive Data) |
| 6 | COM(D-GND) | Communications Common |

TABLE : Digital IO

| Pin No. | Signal | Description |
|---------|----------|--|
| 1 | DIG-IN1 | Digital Input Pin |
| 2 | DIG-IN2 | Digital Input Pin |
| 3 | DIG-IN3 | Digital Input Pin |
| 4 | DIG-OUT1 | Digital Output Pin (output 0V in default) |
| 5 | GND | Ground |
| 6 | VDC-PWR | Power Supply (12V-48V /2A) |

• 2. Revisions

| version | author | date | remarks |
|---------|--------|------------|---|
| V1.0.1 | Tao | 2023-09-10 | |
| V1.0.2 | Tao | 2023-10-31 | Add Pin description |
| V1.0.3 | Tao | 2024-02-26 | 1. Add Digital IO APIs 2. Add Led Light APIs |
| V1.0.4 | Tao | 2024-03-19 | 1.Replace mdbMaster.aar to CM30-HardwareLibrary-1.0.3.aar 2. Add MDB Slave Transport layer APIs 3. Add Physical Key chapter |
| V1.0.5 | Tao | 2024-05-07 | 1. Add USB2COM 2. Digital-OUT supprot negative pulse |
| V1.0.6 | Tao | 2024-12-06 | 1.Add Executive Actuator |
| V1.0.7 | Tao | 2025-01-09 | 1.Move 4.2 to “ Cashless Development Protocol Specification For CM30-V1.0.1.pdf ” |

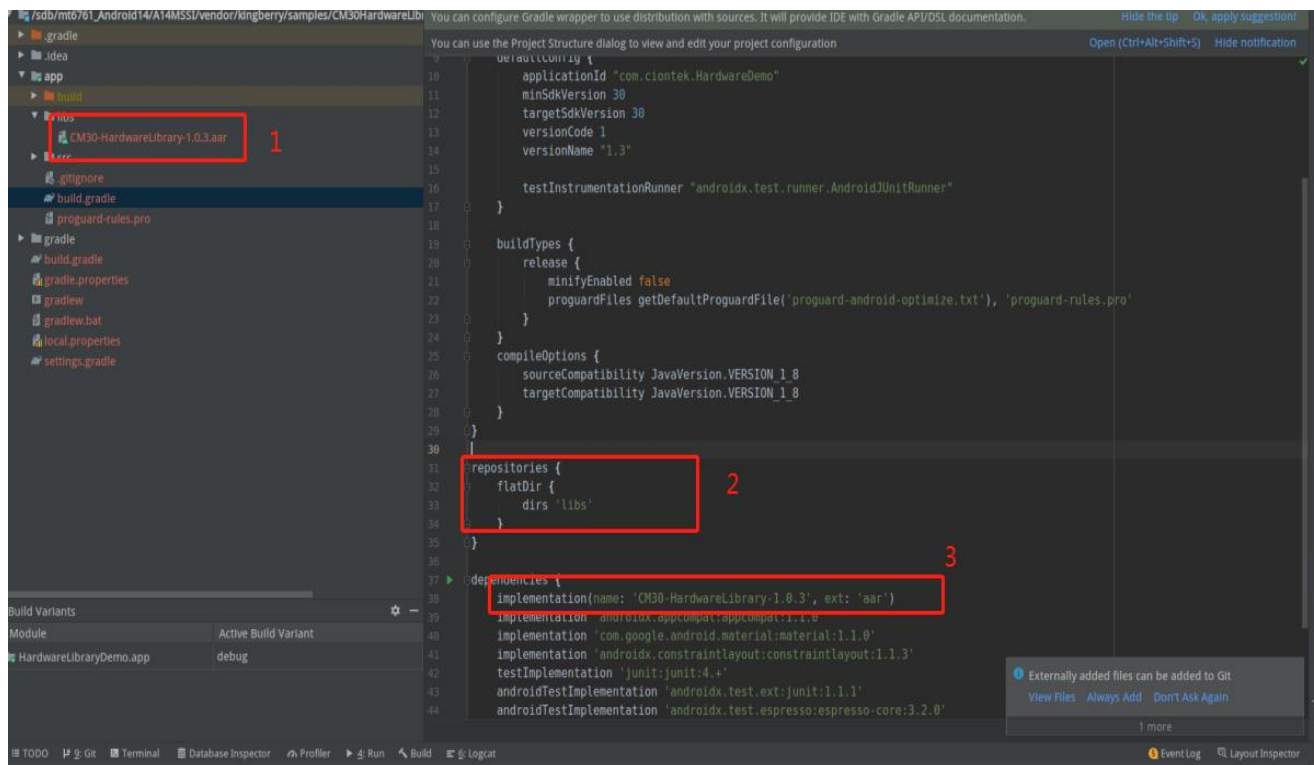
• 3. MDB Master

CM30 is a Vending Machine Controller(VMC) connect to slave devices by MDB Master port that meets Multi-Drop Bus/Internal Communication Protocol. Hereafter is the instruction of all APIs defined by Ciontek for developers to program their own VMC application upon CM30.

For more details please refer to the <MDB/ICP Version 4.2>

3.1 Import the MDB Master SDK for Android Studio

Import the “CM30-HardwareLibrary-1.0.3.aar” to your project, as shown in the picture below:



call example:

```
mMdbMaster = MdbMaster.getInstance();
mMdbMaster.open();
```

3.2. mdbMaster APIs

Class 'MdbMaster' describes all the APIs for MDB Master.

3.2.1 Open

| | |
|------------------------------|---|
| Function prototype | public int open() |
| Parameter description | none |
| Return | 0 - success ; !0 - fail |
| Function description | Open the MDB Master port and power it on. |
| Example | mMdbMaster.open(); |

3.2.2 Close

| | |
|------------------------------|---|
| Function prototype | public int close() |
| Parameter description | none |
| Return | 0 - success ; !0 - fail |
| Function description | Close the MDB Master port and power off |
| Example | mMdbMaster. close (); |

3.2.3 sendAnswer

| | |
|------------------------------|--|
| Function prototype | public int sendAnswer(int respCode) |
| Parameter description | respcode - ACK(0x00) - NAK(0xFF) - RET(0xAA) |
| Return | >0 - success ; <=0 - fail |
| Function description | Send ACK NAK RET to a peripheral device. |
| Example | mMdbMaster. sendAnswer (ACK); |

3.2.4 sendCommand

| | |
|------------------------------|--|
| Function prototype | public int sendCommand(byte[] cmd,int cmdSize,byte[] response) |
| Parameter description | cmd - the command data to be sent cmdSize - the size of cmd, cmdSize <= 36 response - the response data from the peripheral device |
| Return | >0 : size of response data that reply by peripheral <=0 - fail |
| Function description | Send a maximum of 36 bytes of command to the peripheral device and receive response data from peripheral device. |
| Example | mMdbMaster.sendCommand(mCmdBuffer,2,response); |

3.2.5 setLights

| | |
|------------------------------|---|
| Function prototype | public void setLights(int colorRGB,boolean flashing,int onMS) |
| Parameter description | colorRGB - 0xFF0000 (Red LED) - 0x00FF00 (Green LED) - 0x0000FF (Blue LED) Flashing - true (led flashing bright) - false (led Always bright) onMS - when flashing is true,the time of led bright |
| Return | none |
| Function description | Set MDB status LED |
| Example | mMdbMaster.setLights(0xFF0000,true,200); |

3.3 Executive actuator

Class 'mdbExecutive' describes all the APIs for control Pin DIGI-OUT2 & MDB-WAKEUP & DIGI-IN4. See **TABLE : MDB Master/Executive** for pin definition

To get the executive actuator instance:

```
private mdbExecutive mExecutive = mdbExecutive.getInstance();
```

3.3.1 gpio_set_digout_value

DIG-OUT2 pin output 5V high level or 0V low level:

| | |
|------------------------------|--|
| Function prototype | public int gpio_set_digout_value(int value) |
| Parameter description | value : 0 - DIG-OUT2 Pin output 0V low level 1 - DIG-OUT2 Pin output 5V high level |
| Return | 0 - success; !0 - fail |
| Function description | DIG-OUT2 pin output 5V high level or 0V low level |
| Example | mExecutive.gpio_set_digout_value(1); |

3.3.2 gpio_set_wakeup_value

MDB-WAKEUP pin output 5V high level or 0V low level:

| | |
|------------------------------|--|
| Function prototype | public int gpio_set_wakeup_value(int value) |
| Parameter description | value : 0 - MDB-WAKEUP Pin output 0V low level 1 - MDB-WAKEUP Pin output 5V high level |
| Return | 0 - success; !0 - fail |
| Function description | MDB-WAKEUP pin output 5V high level or 0V low level |
| Example | mExecutive.gpio_set_wakeup_value(1); |

3.3.3 gpio_get_digin_value

Detects the state of the DIG-IN4 Pin:

| | |
|------------------------------|--|
| Function prototype | public int gpio_get_digin_value() |
| Parameter description | none |
| Return | 0 - the state of the DIG-IN4 Pin is 0V 1 - the state of the DIG-IN4 Pin is 5V |
| Function description | Detect the state of the DIG-IN4 Pin |
| Example | int state = mExecutive.gpio_get_digin_value(); |

3.4 Return code

Mdb master return code

| Return code | value | description |
|--------------------|--------------|--------------------------------|
| success | 0 | successfully |
| err_fail | -1 | failure |
| err_write | -2 | mdb master port send an error |
| err_read | -3 | mdb master port receives error |
| err_chksum | -4 | check sum error |
| err_nonresp | -5 | peripheral non-response |

• 4. MDB Slave

CM30 is a slave device connect to VMC by MDB Slave port that meets Multi-Drop Bus/Internal Communication Protocol.

CM30 supported two levels MDB Slave APIs, One is the transport layer APIs that allow developers can program all MDB slave peripheral application(Cashless Device/Communications Gateway/Bill Validator/Coin Changer) upon CM30. Another is high level APIs of Cashless Device that make developers program their own Cashless application more easy.

4.1. MDB Slave Transport Layer APIs

By transport layer APIs, Developers can program all the MDB slave peripheral application(Cashless Device/Communications Gateway/Bill Validator/Coin Changer) upon CM30.

Import the “CM30-HardwareLibrary-1.0.3.aar” to your project , For more details please refer to the part 3.1.

Class ‘MdbSlave’ describes all the MDB Slave transport layer APIs .

call example:

```
MdbSlave mMdbSlave = MdbSlave .getInstance();  
mMdbSlave.open();
```

4.1.1 open

Open the MDB Slave port and power on.

| | |
|------------------------------|--------------------------------------|
| Function prototype | public int open() |
| Parameter description | none |
| Return | 0 - success ; !0 - fail |
| Function description | power on and open the MDB slave port |
| Example | mMdbSlave.open() |

4.1.2 close

Close the MDB Slave port and power off.

| | |
|---------------------------|--------------------|
| Function prototype | public int close() |
|---------------------------|--------------------|

| | |
|------------------------------|--|
| Parameter description | none |
| Return | 0 - success ; !0 - fail |
| Function description | power off and close the MDB slave port |
| Example | mMdbSlave.close() |

4.1.3 receiveCommand

Receive a maximum of 36 bytes of command from the VMC

| | |
|------------------------------|--|
| Function prototype | public int receiveCommand(byte[] buffer) |
| Parameter description | Buffer: the receive buffer |
| Return | >0 - success ; <=0 - fail |
| Function description | receive a maximum of 36 bytes of command from the VMC |
| Example | byte[] buffer = new byte[36]; int ret = mMdbSlave.receiveCommand(buffer); |

4.1.4 sendAnswer

Send ACK or NAK or RET to VMC.

| | |
|------------------------------|--|
| Function prototype | public int sendAnswer(int respcode) |
| Parameter description | respcode - ACK(0x00) - NAK(0xFF) - RET(0xAA) |
| Return | >0 - success ; <=0 - fail |
| Function description | Send ACK or NAK or RET to VMC |
| Example | mMdbSlave.sendAnswer(ACK); |

4.1.5 sendResponseData

Send a maximum of 36 bytes of response data to the VMC.

| | |
|------------------------------|---|
| Function prototype | public int sendResponseData(byte[] data,int dataSize,int[] respcode) |
| Parameter description | data - the response data to be sent dataSize - the size of data, dataSize <= 36 respcode - the response code reply by VMC for current response data |
| Return | >0 - success ; <=0 - fail |
| Function description | Send a maximum of 36 bytes of response data to the VMC. |
| Example | mMdbSlave.sendResponseData(responseData,2,respcode); |

4.2. MDB Cashless Device APIs

Please refer to another document “Cashless Development Protocol Specification For CM30-V2.0.1.pdf”.

• 5. RS232

CM30 has two channel RS232 ports(channel A and channel B).

Channel A and channel B cannot be used simultaneously.

APIs for RS232 in the library “CM30-HardwareLibrary-1.0.3.aar”, Import the “CM30-HardwareLibrary-1.0.3.aar” to your project , For more details please refer to the part 3.1.

Class ‘RS232’ describes all the APIs for RS232

5.1 RS232Open

| | |
|------------------------------|--|
| Function prototype | public int RS232Open(int channel, int baudrate, int size, int stop, char parity, char cflow) |
| Parameter description | channel: 0 - channel A 1 -channel B baudrate: the baudrate of RS232 size : data bits of RS232 stop : stop bits of RS232 parity : parity bit of RS232 cflow : Control options RS232 |
| Return | 0 - success; !=0 - fail |
| Function description | Open and initialize the RS232 channel that was selected. |
| Example | mRS232Port.RS232Open(0,115200,8,1,'N','N'); |

5.2 RS232Close

| | |
|------------------------------|-------------------------------------|
| Function prototype | public int RS232Close() |
| Parameter description | none |
| Return | 0 - success; !=0 - fail |
| Function description | Close and power off the RS232 port. |
| Example | mRS232Port.RS232Close (); |

5.3 RS232Write

| | |
|------------------------------|-------------------------------------|
| Function prototype | public int RS232Write(byte[] data) |
| Parameter description | data that to be sent |
| Return | 0 - success; !=0 - fail |
| Function description | Send data by RS232 port. |
| Example | mRS232Port.RS232Write(send); |

5.4 RS232Read

| | |
|------------------------------|--|
| Function prototype | public int RS232Read(byte[] buffer,int bufLen,int timeout) |
| Parameter description | buffer - the buffer for data from rs232 bufLen - the length of the buffer timeout - timeout for read, unit: ms |
| Return | >0 : the counts read from serial port <0: read fail -1: fail -2: uninitialized -3: parameter error -4: timeout -5: init uart port error -6: read error -7: write error |

| | |
|-----------------------------|--|
| | |
| Function description | Read data from RS232 port. |
| Example | mRS232Port.RS232Read(buffer, 1024, 100); |

• 6. Digital IO

CM30 provide one 'DIG-OUT' output pin and three 'DIG-IN' input pin for developer.

APIs for Digital IO in the library "CM30-HardwareLibrary-1.0.3.aar", Import the "CM30-HardwareLibrary-1.0.3.aar" to your project , For more details please refer to the part 3.1.

Class 'DigitalIO' describes all the APIs for Digital IO.

6.1 digital_out_set_value

DIG-OUT Pin output high or low level:

| | |
|------------------------------|--|
| Function prototype | public int digital_out_set_value(int value) |
| Parameter description | value: 0 - DIG-OUT Pin output 0V low level 1 - DIG-OUT Pin output 5V high level |
| Return | 0 - success; !=0 - fail |
| Function description | DIG-OUT pin output 5V high level or 0V low level |
| Example | mDigitalIO.digital_out_set_value(1); |

6.2 digital_out_pulse

DIG-OUT Pin output pulse:

| | |
|------------------------------|--|
| Function prototype | public int digital_out_pulse(int direction,int count,int period,int width) |
| Parameter description | direction: PULSE_DIR_POSITIVE output positive pulse PULSE_DIR_NEGATIVE output negative pulse count: The number of output pulses period: Pulse period (ms) width: Pulse width (ms) |
| Return | 0 - success; !=0 - fail |
| Function description | DIG-OUT pin output pulse |

| | |
|----------------|---|
| Example | <code>mDigitalIO.digital_out_pulse(DigitalIO.PULSE_DIR_NEGATIVE,100,100,50);</code> |
|----------------|---|

6.3 digital_in_get_value

Detects the state of the DIG-IN Pins:

| | |
|------------------------------|---|
| Function prototype | <code>public int digital_in_get_value(int gpio)</code> |
| Parameter description | gpio: 1 - Pin DIG-IN1 2 - Pin DIG-IN2 3 - Pin DIG-IN3 |
| Return | 0 - the state of the DIG-IN Pin is 0V 1 - the state of the DIG-IN Pin is 5V |
| Function description | Detect the state of the DIG-IN Pins |
| Example | <code>int value = mDigitalIO.digital_in_get_value(1);</code> //get the state of DIG-IN1 |

6.4 digital_in_request_irq

Set DIG-IN Pins request IRQ interrupt:

| | |
|------------------------------|---|
| Function prototype | <code>public int digital_in_request_irq(int gpio,int flags)</code> |
| Parameter description | gpio: 1 - Pin DIG-IN1 2 - Pin DIG-IN2 3 - Pin DIG-IN3 flags: IRQF_TRIGGER_RISING - Rising edge trigger IRQF_TRIGGER_FALLING - Falling edge trigger |
| Return | 0 - success; !=0 - fail |
| Function description | Set DIG-IN request IRQ interrupt. |
| Example | <code>mDigitalIO.digital_in_request_irq(1,DigitalIO.IRQF_TRIGGER_RISING);</code> |

6.5 registerDigitalIOInterruptMonitor

When you request an IRQ interruption, you also need to register a monitor for interruption.

| | |
|------------------------------|---|
| Function prototype | <code>public boolean registerDigitalIOInterruptMonitor(DigitalIOInterruptMonitor callback)</code> |
| Parameter description | DigitalIOInterruptMonitor callback: An interrupt monitor |

| | |
|-----------------------------|--|
| Return | 0 - success; !0 - fail |
| Function description | register a monitor for interrupt |
| Example | <pre> mDigitalIO.registerDigitalIOInterruptMonitor(new DigitalIOInterruptMonitor() { @Override public void onDigitalIO1() { } @Override public void onDigitalIO2() { } @Override public void onDigitalIO3() { } }); </pre> |

Important note: Please don't do any time-consuming work in the **DigitalIOInterruptMonitor**

6.6 unregisterDigitalIOInterruptMonitor

Unregister the monitor

| | |
|------------------------------|---|
| Function prototype | public void unregisterDigitalIOInterruptMonitor() |
| Parameter description | null |
| Return | void |
| Function description | Unregister the monitor |
| Example | mDigitalIO.unregisterDigitalIOInterruptMonitor(); |

Important note: If an **DigitalIOInterruptMonitor** registered, **unregisterDigitalIOInterruptMonitor** must be called when finish your application.

6.7 Return code

Digital IO Return code

| Return code | value | description |
|---------------|-------|-----------------|
| success | 0 | successfully |
| err_fail | -1 | failure |
| err_param | -2 | parameter error |
| err_gpio_busy | -3 | gpio busy |

- **7. Led Light**

APIs for Led Light in the library “CM30-HardwareLibrary-1.0.3.aar”, Import the “CM30-HardwareLibrary-1.0.3.aar” to your project , For more details please refer to the part 3.1.

Class ‘Light’ describes all the APIs for Led Light.

7.1 getLightInstance

| | |
|------------------------------|--|
| Function prototype | public static Light getLightInstance() |
| Parameter description | null |
| Return | Led Light instance |
| Function description | get the Led Light instance |
| Example | private Light mLight = Light.getLightInstance(); |

7.2 setLights

| | |
|------------------------------|---|
| Function prototype | public void setLights(int colorRGB,boolean flashing,int onMS) |
| Parameter description | colorRGB - 0xFF0000 (Red LED) - 0x00FF00 (Green LED) - 0x0000FF (Blue LED) Flashing - true (led flashing bright) - false (led Always bright) onMS - when flashing is true,the time of led bright |
| Return | none |
| Function description | Set the Led status |
| Example | mLight .setLights(0xFF0000,true,200); |

7.3 destroy

| | |
|------------------------------|-----------------------|
| Function prototype | public void destroy() |
| Parameter description | null |
| Return | void |

| | |
|-----------------------------|--------------------------------|
| Function description | Release the Led Light instance |
| Example | mLight.destroy(); |

• 8. Physical Key

CM30 has a 'RESET' key and a 'SERVICE' key. Long press RESET key CM30 will reboot. And the SERVICE key is reserved for developers to monitor.

The keyCode of 'SERVICE' key is KeyEvent.KEYCODE_F1

Developers implements the View.OnClickListener to listen key event, For example:

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if(keyCode == KeyEvent.KEYCODE_F1){
        Log.d(TAG,"onKeyDown: ServiceKey Down ");
        return true;
    }
    return super.onKeyDown(keyCode,event);
}
```

```
@Override
public boolean onKeyUp(int keyCode, KeyEvent event) {
    if(keyCode == KeyEvent.KEYCODE_F1){
        Log.d(TAG,"onKeyUp: ServiceKey UP ");
        return true;
    }
    return super.onKeyUp(keyCode,event);
}
```

• 9. USB2COM

CM30 supports USB works on Serial mode. When USB2COM mode is enabled and connected to the host side (PC or other USB host devices) with a USB cable, Developers can send/receive serial data.

Notes: USB2COM works on the USB device mode, you need to switch USB to device mode and enable USB2COM in "Settings-Connect devices".

On Windows, when CM30 connected to PC, you should see "Com*" listed in the "Device Manager" (under

"Control Panel", "System", "Hardware").

APIs for USB2COM in the library “CM30-HardwareLibrary.aar”, Import the “CM30-HardwareLibrary.aar” to your project , For more details please refer to the part 3.1.

Class ‘USBCom’ describes all the APIs for USB2COM

9.1 Open

| | |
|------------------------------|--|
| Function prototype | public int Open(int baudrate,int size, int stop, char parity, char cflow) |
| Parameter description | baudrate : the baudrate of the USB Com size : data bits of the USB Com stop : stop bits of the USB Com parity : parity bit of the USB Com cflow : Control options |
| Return | 0 - success; !0 - fail |
| Function description | Open the USB Com port. |
| Example | mUSBCom.Open(9600,8,1,'N','N'); |

9.2 Close

| | |
|------------------------------|-------------------------|
| Function prototype | public int Close() |
| Parameter description | none |
| Return | 0 - success; !0 - fail |
| Function description | Close the USB Com port. |
| Example | mUSBCom.Close(); |

9.3 Write

| | |
|------------------------------|--------------------------------|
| Function prototype | public int Write(byte[] data) |
| Parameter description | data that to be sent |
| Return | 0 - success; !0 - fail |

| | |
|-----------------------------|--------------------------------|
| Function description | Send data by the USB Com port. |
| Example | mUSBCom.Write(send); |

9.4 Read

| | |
|------------------------------|---|
| Function prototype | public int Read(byte[] buffer,int bufLen,int timeout) |
| Parameter description | buffer - the buffer for data from the USB Com port bufLen - the length of the buffer timeout - timeout for read, unit: ms |
| Return | >0 : the counts read from the USB Com port <0: read fail -1: fail -2: uninitialized -3: parameter error -4: timeout -5: init uart port error -6: read error -7: write error |
| Function description | Read data from the USB Com port |
| Example | mUSBCom.Read(buffer, 1024, 100); |