GV850 Software Development Guide

Version: 1.01

International Telematics Solutions Innovator

www. queclink .com

Document Title	GV850 Software Development Guide
Version	1.01
Date	2023-11-27
Status	Released

General Notes

Queclink offers this information as a service to its customers, to support application and engineering efforts that use the products designed by Queclink. The information provided is based upon requirements specifically provided to Queclink by the customers. Queclink has not undertaken any independent search for additional relevant information, including any information that may be in the customer's possession. Furthermore, system validation of this product designed by Queclink within a larger electronic system remains the responsibility of the customer or the customer's system integrator. All specifications supplied herein are subject to change.

Copyright

This document contains proprietary technical information which is the property of Queclink. Copying of this document, distribution to others or using or communication of the contents thereof is forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of a patent grant or the registration of a utility model or design. All specifications supplied herein are subject to change without notice at any time.

Copyright © Queclink Wireless Solutions Co., Ltd. 2023

Queclink Confidential

Contents

Queclink Confidential

0. Revision History

Version	Date	Author	Description of Change
1.00	2023-09-25	Alex Liao	Initial
1.01	2023-11-21	Alex Liao	Added more information to make the file more complete.

Queclink Confidential

1. Overview

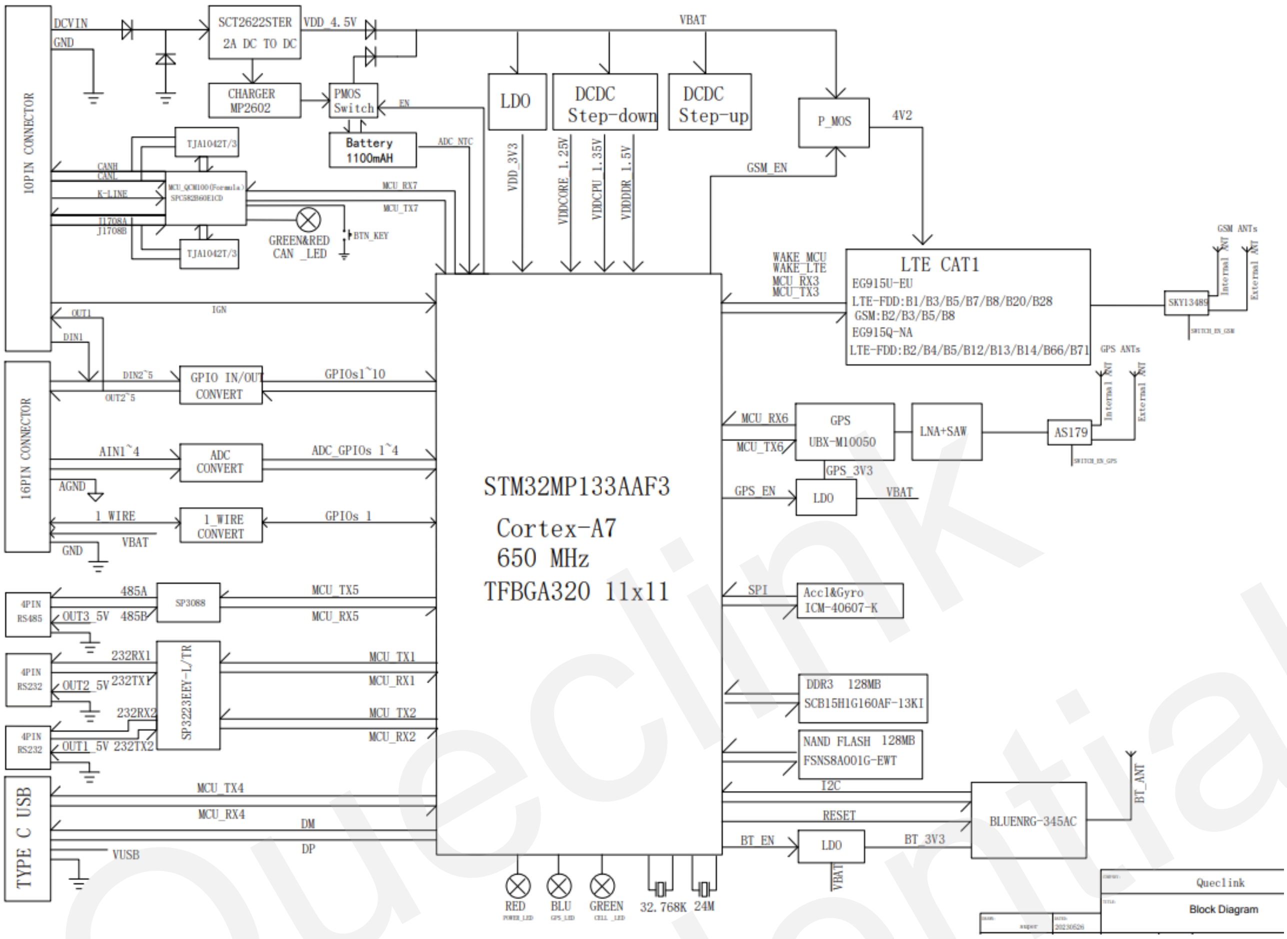
MPU	STM32MP133A
RAM	128MB, DDR3 or above
FLASH memory	128MB SPI or above
Status LEDs	1 x Power, 1 x CEL, 1 x GNSS LED, 1 x CAN/Tachograph
SIM	1 x SIM card slot or eSIM
Modem	Support Cat 1 LTE-FDD: B1/B3/B5/B7/B8/B20/B28 GSM: B2/B3/B5/B8
RS232 or RS485	2 x RS232, 300-115200 baud rate 1 x RS485, 300-115200 baud rate/Half Duplex (2 wires)
I/O	1 x positive trigger input for ignition detection 5 x negative trigger inputs 4 x analog input (0-32V) 5 x digital output, open drain, 150mA max drive current 1 DC 5V output for temperature sensor 3.5V outputs for external accessories
CAN	CAN1H/CAN1L, support reading CAN bus data in heavy (J1939/FMS) and light vehicle CAN2H/CAN2L, support reading and download tachograph data, support reading CAN data in J1708 and OBDII
K-Line	Connect D8 of tachograph for live data reading
Type-C USB	Used for configuration, upgrade and debug
1-Wire Interface	Support 1-wire temperature sensor and iButton driver ID
GNSS	u-box all-in-one GNSS receiver, support GPS, Glonass, Galileo, Beidou
BLE	BLE5.2
Battery	Li-Polymer, 250mAh
G-sensor	6-axis motion, motion detection, harsh driving detection, shock detection
Reset button	Reset button to reset system
Cellular Antenna	Internal or external
GNSS Antenna	Internal or external
BLE Antenna	Internal

Queclink Confidential

Sleep Current	< 10mA
Firmware/configuration	
Operating system	Linux OS, Kernel 5.15.67
Power	
Connector	Pin connector
Input voltage range	8 – 32 VDC, reverse polarity protection; surge protection >31 VDC 10us max
Power consumption	5W (average)
Physical Specification	
Dimensions	123*80*21mm (L*W*H)
Weight	150g
Mounting options	Flat surface placement
Operating Environment	
Operating temperature	-30 °C to 75 °C
Operating humidity	10% to 90% RH non-condensing
Ingress Protection Rating	IP30

Queclink Confidential

The hardware block diagram is as following:



Two construction methods, buildroot and yocto, and corresponding SDK source codes, are provided. The positioning from Linux of these two construction methods differs (though both are commonly used in embedded systems, but there are differences in efficiency and usage methods):

- Buildroot, which builds a more streamlined and simple system and is suitable for devices with limited hardware resources (mainly flash);
- Yocto, which builds a system with rich features and supports more complete hardware, including UI, audio and video software stacks, requiring a larger flash size.

Queclink Confidential

2. Platform Development

At present, source codes for building systems based on buildroot are provided, which can build and package complete system images.

2.1. devicetree

The first step in developing STM32MP1 platform devices is to adapt a devicetree based on its hardware. Moreover, because the devicetree is used in each module of bootchain, it is a complex and cumbersome operation to ensure that each module obtains the correct devicetree during compilation. Therefore, ST has provided the STM32CubeMX tool to provide visualization, assistance, and configuration wizards that can automatically generate the devicetree required by each module. The provided buildroot source codes already contain the adapted devicetree.

2.2. bootchain

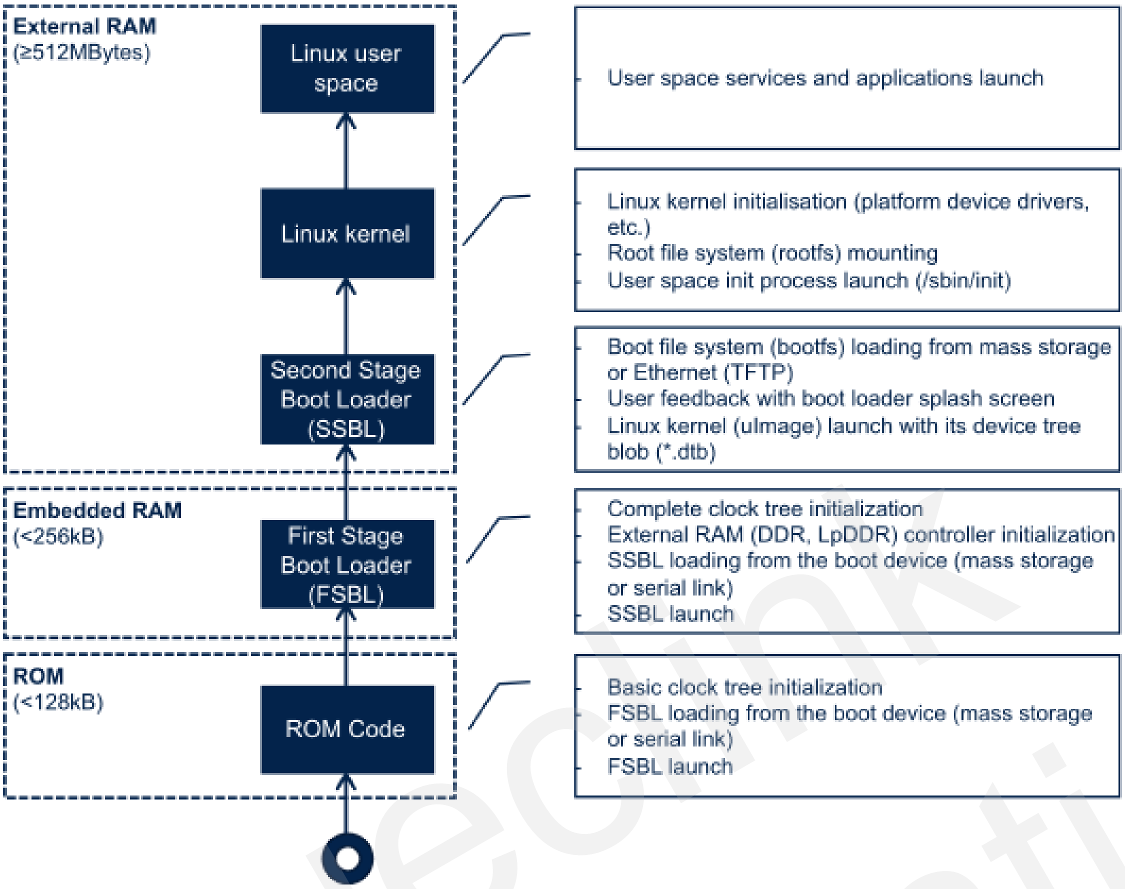
The STM32MP133 platform is based on the ARM Cortex-A7 architecture, and the boot process is similar to other ARM architectures. It is mainly divided into the following stages:

ROM code
FSBL (First stage bootloader) TF-A
SSBL (Second-stage bootloader), u-boot

Please search “boot chain” in following link to get more information.

- https://wiki.stmicroelectronics.cn/stm32mpu/wiki/Main_Page

Queclink Confidential



After understanding the startup process of STM32 ARM, it's known that the actual module relationship is:

TF-A->OP-TEE->U-BOOT->Linux Kernel

2.3. Compilation Method

To use the Buildroot, there must be a Linux distribution installed on the workstation. Any reasonably recent Linux distribution (Ubuntu, Debian, Fedora, Redhat, OpenSuse, etc.) will work fine. Then, a small set of packages needs to be installed as described in the System Requirements section of Buildroot Manual.

For Debian/Ubuntu distributions, use the following command to install the necessary packages:

```
$ sudo apt-get install -y debianutils sed make binutils build-essential gcc g++ \
            bash patch gzip bzip2 perl tar cpio unzip rsync file bc git \
            wget python3 libssl-dev libncurses-dev
```

After finishing installation, extract source tarball that is provided:

```
$ tar zxvf GV850_buildroot_dd981da1.tar.gz
```

Go to the Buildroot directory:

```
$ cd GV850_buildroot_dd981da1/buildroot/
```

And then, configure the system you want to build by using the defconfigs provided in this BR2_EXTERNAL tree.

```
$ make BR2_EXTERNAL=../buildroot-external-st st_stm32mp133a_queclink_GV850CEU_defconfig
```

There are two pieces of information are provided:

1. The path to BR2_EXTERNAL tree, which is provided side-by-side to the Buildroot repository.

2. The name of the Buildroot configuration.

If there is the need to further customize the Buildroot configuration, please run 'make menuconfig', but for the first build, it is recommended to keep the configuration unchanged so that it can be verified that everything is working.

Start the build:

```
$ make V=s
```

It might take between 30 and 60 minutes depending on the configuration that is chosen and how powerful the machine is. All software packages for building the entire Linux system for the STM32MP1 platform (e.g. cross-compilation toolchain, firmware, bootloader, Linux kernel, root filesystem) are already included, no downloading is needed unless default configuration is customized.

Buildroot might need to be authorized to root (or sudo) in order to compile some packages (related to Python 3) properly. If some permission failures are met, please retry:

```
$ sudo make V=s
```

When the building is done, it will output images in the directory below, including u-boot, kernel, rootfs binary files.

```
$ cd output/images
```

Following files in this directory are necessary for flashing, please copy and prepare for flashing.

```
├── fip.bin
├── flash.tsv
├── metadata.bin
├── rootfs.ubi
└── tf-a-stm32mp133a-gv850ceu-mx.stm32
```

2.4. Programming

The device supports both USB OTG programming and OTA firmware updating.

USB OTG

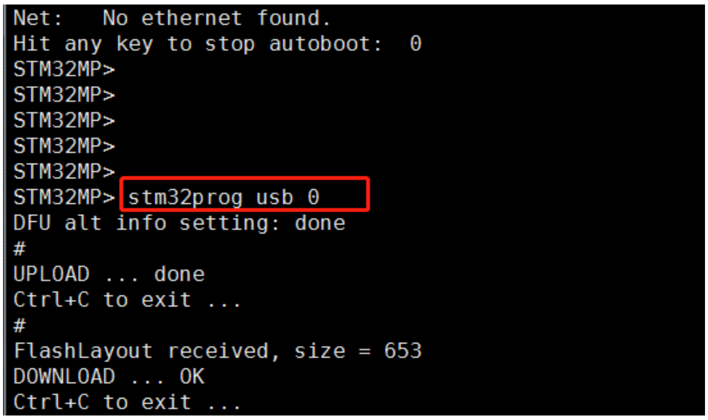
After successfully building with Buildroot, the complete files required for programming can be obtained.

```
├── fip.bin                                      // FIP
├── flash.tsv                                // Program partitions configuration table
├── metadata.bin
├── rootfs.ubi                                   // Including kernel and file system rootfs
└── tf-a-stm32mp133a-gv850ceu-mx.stm32      // TF-A
```

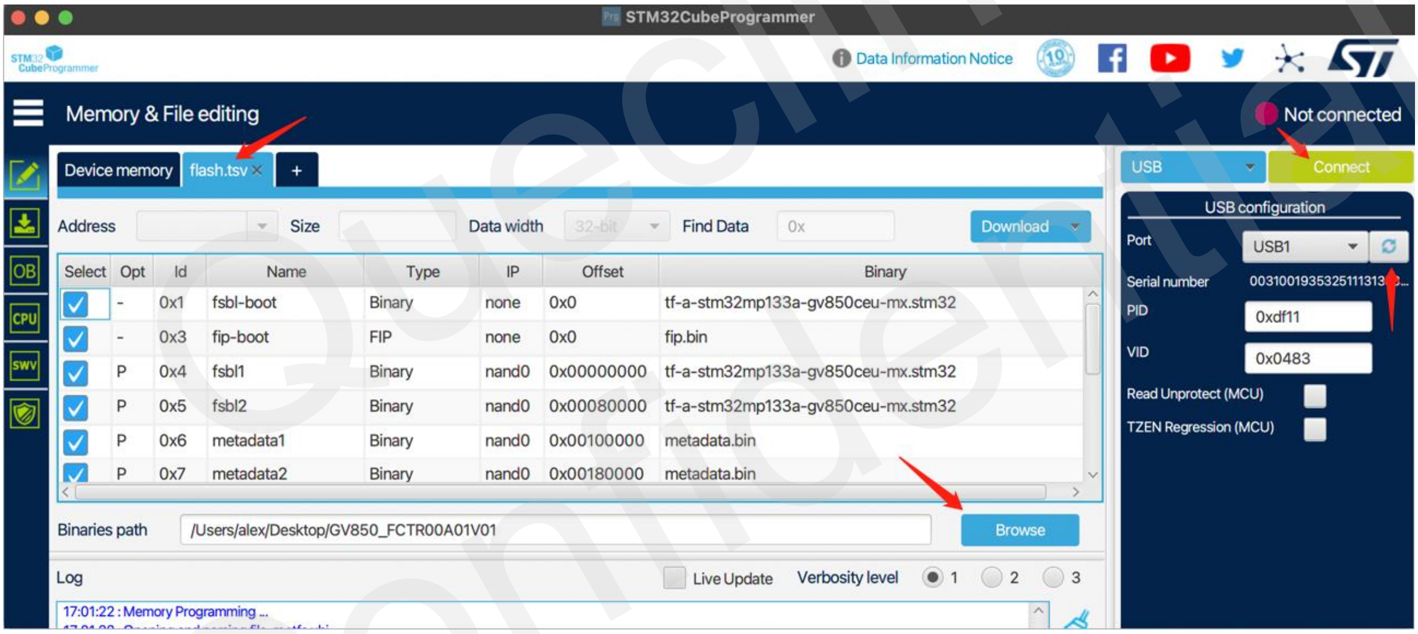
The device first enters DFU mode. And then use the STM32CubeProgrammer tool to erase and programm the device. The method and steps are as follows:

1. Use the USB+UART 2-in-1 cable provided along with the device, open the COM device on a PC using the UART tool, and the baud rate is 115200bps;
2. Power on the device, the COM starts printing the startup log, and then quickly press any key on the keyboard. The startup process will be interrupted and it requires to enter the u-Boot command. Then, enter the following command to enter DFU mode;

```
STM32MP> stm32prog usb 0
```



3. Connect the USB of the cable to the PC, click the right button of the mouse to click refresh, after automatically scanning and finding the device that has entered DFU mode, and then click "Connect"
4. Select "Open file" to load the flash. tsv file from the released firmware, and note that select the correct path for "Browse";
3. Click "Download" to start programming. After successful programming, power off the device, unplug and reinsert the USB Type-C cable, power on the device, and the device enters the boot process.



OTA

Still under development.

Queclink Confidential

3. Application Development

In order to facilitate developers to familiarize themselves with the platform, example code and software packages of testing programs are provided. Please Compile it using the following command:

```
$ make queclink-dirclean
$ make queclink
```

3.1. Debugging Tool

GV850 only has USB and serial ports. During the development and debugging process, it is inevitable to frequently modify binary programs or scripts. In order to avoid using STM32Cube Programmer for programming, which is an inefficient method, serial or USB methods can be used.

rz, sz —— A transmission tool that supports the ZMODEM/YMODEM/XMODEM protocol and can upload/download files to the device through client-end software. The transmission efficiency depends on the physical connection of the transmission, such as the serial port baud rate or USB speed rate;

Ethernet —— After enabling kernel configuration for STM32MP1 platform, the device can be plugged into the host PC through the OTG USB port. The device will be virtualized as an RNDIS network device, and a network device named "usb0" will also be generated inside the device system. After configuring the same network segment IP address, the two can communicate.

Reference IP address configured to the device:

```
$ ifconfig usb0 192.168.0.1 netmask 255.255.255.0
```

It can be correctly identified as a network device on virtual machine ubuntu16

```
avalon@avalon-virtual-machine:~$ lsusb
Bus 001 Device 007: ID 0525:a4a2 Netchip Technology, Inc. Linux-USB Ethernet/RNDIS Gadget
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
avalon@avalon-virtual-machine:~$
```

After configuring the IP address, network communication can be established.

```
$ ifconfig ens35u1i1 192.168.0.2 netmask 255.255.255.0
```

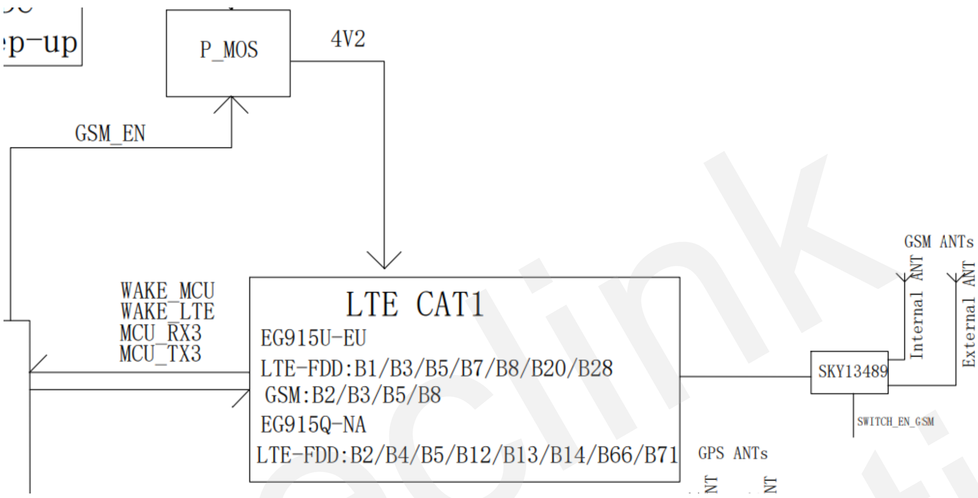
Note: If due to driver issues, the virtual network device cannot be correctly identified. Please try to install/update the driver mod-rndis-driver-windows.zip.

Queclink Confidential

4. Interface and Driver

4.1. LTE

Module model: EG915UEU, which is not in network card mode, but in uart module mode. It uses USART3, which corresponds to /dev/ttySTM3. The schematic diagram is as follows:



Reference testing commands

Set the baud rate to 115200bps and remove the incrnl attribute to avoid automatically converting input characters \r to \n. Remove the isig icanon echo echoe attribute to avoid output causing incorrect module command format +CME ERROR: 58 error.

$ stty -F /dev/ttySTM3 ispeed 115200 ospeed 115200 cs8 -icrnl -isig -icanon -echo -echoe

Receive module uart output,

$ cat /dev/ttySTM3

PA15 module power supply enable output high,

$ gpioset 0 15=1

PF5 module startup signal,

$ gpioset 5 5=1

$ gpioset 5 5=0

Receive the module startup URC message,

RDY

Turn off echo,

$ echo "ATE0" > /dev/ttySTM3

Internal antenna or external antenna can be selected, select internal antenna,

$ gpioset 4 2=1

Or select external antenna,

$ gpioset 4 2=0

Queclink Confidential

WAKE_ LTE (DTR pin PE14) controls the sleep of the module, high level allows sleep, and low level wakes up the module,

```
$ gpioset 4 14=0
```

Query the DTR pin status via command, if it is 0, sleep is not allowed,

```
$ echo "AT+QGPIOR=25" > /dev/ttySTM3
+QGPIOR: 0

OK
```

DTR pin output high level,

```
$ gpioset 4 14=1
```

Query the DTR pin status via command, if it is 1, sleep is allowed

```
$ echo "AT+QGPIOR=25" > /dev/ttySTM3
+QGPIOR: 1

OK
```

Send the AT+QSCLK=1 command to enter sleep,

```
$ echo "AT+QSCLK=1" > /dev/ttySTM3
OK
```

Sending any AT command will wake up the module, but at the appropriate time, it will enter sleep again unless the DTR pin output is at low level or the sleep function is turned off using the AT+QSCLK=0 command.

The module can notify the MPU through the level change of the WAKE_MCU (RI pin PD3). Due to the rapid level change, it is not possible to accurately obtain it using gpioget. Therefore, the gpiomon tool can be used for monitoring it.

For example, using command to turn off the module,

```
$ echo "AT+QPOWD" > /dev/ttySTM3
POWERED DOWN
```

Monitoring receives GPIO level change events,

```
$ gpiomon 3 3
event: FALLING EDGE offset: 3 timestamp: [       2582.179763702]
event:    RISING EDGE offset: 3 timestamp: [       2582.300251859]
event: FALLING EDGE offset: 3 timestamp: [       2582.618285921]
event:    RISING EDGE offset: 3 timestamp: [       2582.618430593]
```

Use the provided example_modem_at tool for command testing, as detailed in the "Example of Codes" section.

The LTE module can serve as a wake-up source for system sleep, as detailed in the "System Sleep" section.

The following demonstrates the process of how to connect to the network, send and receive TCP data.

Check for correct SIM card reading,

```
$ echo "AT+CPIN?" > /dev/ttySTM3
+CPIN: READY

OK
```

Check CS status,

```
$ echo "AT+CREG?" > /dev/ttySTM3
```

Queclink Confidential

```
+CREG: 0,1

OK
```

Attach PS domain,

```
$ echo "AT+CGATT=1" > /dev/ttySTM3
OK
$ echo "AT+CGATT?" > /dev/ttySTM3
+CGATT: 1

OK
```

Activate PDP,

```
$ echo "AT+QIACT=1" > /dev/ttySTM3
OK
```

Check the PDP status and obtained IP address,

```
$ echo "AT+QIACT?" > /dev/ttySTM3
+QIACT: 1,1,3,"10.162.247.73","2408:8456:3040:AB7:1:1:A0D9:4891"

OK
```

Ping domain name to check network connectivity,

```
$ echo "AT+QPING=1,\"www.baidu.com\"" > /dev/ttySTM3
OK

+QPING: 0,"157.148.69.74",64,313,255
+QPING: 0,"157.148.69.74",64,61,255
+QPING: 0,"157.148.69.74",64,61,255
+QPING: 0,"157.148.69.74",64,50,255
+QPING: 0,4,4,0,50,313,87
```

Open socket, using 218.17.50.142:971 server/port as the example,

```
$ echo "AT+QIOPEN=1,0,\"TCP\",\"218.17.50.142\",971,0,0" > /dev/ttySTM3
OK

+QIOPEN: 0,0
```

Check the status of the socket and confirm that it is connected,

```
$ echo "AT+QISTATE?" > /dev/ttySTM3
+QISTATE: 0,"TCP","218.17.50.142",971,0,2,1,0,0,"uart1"

OK
```

Send the test string '12345' in HEX format,

```
$ echo "AT+QISENDEX=0,\"3132333435\"" > /dev/ttySTM3
```

```
SEND OK
```

The server responds with data '67890', and the module will notify the module with a URC message upon receiving the data,

```
+QIURC: "recv",0
```

At this point, the received data can be read from the cache and the actual length and data will be returned,

```
$ echo "AT+QIRD=0,1500" > /dev/ttySTM3
+QIRD: 5

67890


OK
```

Close socket,

```
$ echo "AT+QICLOSE=0" > /dev/ttySTM3
OK
```

Check the status of the socket and confirm that it is closed,

```
$ echo "AT+QISTATE?" > /dev/ttySTM3
OK
```

4.2. Watchdog

GV850 adopts an external independent hardware watchdog.

Pin Name	Description	Remarks
PI 7	Watchdog enable IO	output high, enable watchdog output low, disable watchdog
PG 14	Restart watchdog IO	Flip the level within 1.7s, otherwise a reset will be triggered.

The software watchdog restart is implemented through a qdog driver and a sysfs interface is provided to enable and disable the watchdog,

```
$ lsmod | grep qdog
qdog                    16384    0
```

Turn on watchdog and restart the watchdog automatically,

```
echo 1 > /proc/qlwatchdog_enabled
```

Turn off watchdog,

```
echo 0 > /proc/qlwatchdog_enabled
```

4.3. RTC

STM32MP133 has built-in RTC, device/dev/rtc0, and can be set and obtained through the system's built-in hwclock tool. When the system starts, it will be loaded and set as the local time of the system. Reference command:

Query the current system time,

```
$ date
Wed Jan  5 03:19:16 UTC 2000
```

Set the system time to local time

```
$ date -s "2023-09-27 14:26:30"
Wed Sep 27 14:26:30 UTC 2023
```

Set the system time to RTC

```
$ hwclock –w
```

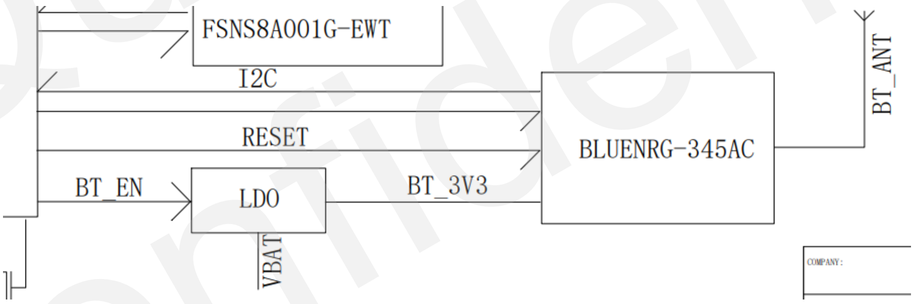
Read time from RTC

```
$ hwclock -r
Wed Sep 27 14:27:12 2023  0.000000 seconds
```

The RTC can serve as a wake-up source for system sleep, as detailed in the "System Sleep" section.

4.4. BLE

Module model: BlueNRG-345AC, connected through I2C bus. STM32MP133 platform reads and writes from I2C bus 0 through/dev/i2c-0 device.



The reference testing commands are as follows:

PE15 power supply enable output high,

```
$ gpioset 4 15=1
```

PG7 is used to reset BLE,

```
$ gpioset 6 7=1
```

Scan I2C bus 0,

```
$ i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- 3e --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Queclink Confidential

```
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

After scanning the slave device on the I2C bus, the device boot message can be read. The BLE module serves as the I2C slave device with address 0xBE and register address 0x01, and reads 220 bytes each time. The command/protocol description is detailed in the document "BLE100 @ Bluetooth Internal Protocol".

```
$ i2ctransfer -y 0 w1@0x3e 0x01 r220
```

The provided example_ble tool can also be used for command testing, as detailed in the "Example of Codes" section. The BLE module is developed by Queclink itself. The command/protocol description is detailed in the document "BLE100 @ Bluetooth Internal Protocol".

The sleep of the BLE module can be controlled, PH12 output high level allows sleep and low level wakes up the module,

```
$ gpioset 7 12=1
```

BLE Module events can be notified to the MPU through the PG4 pin, such as sending the command AT+F=12 to the BLE module, which will wake up the MPU,

```
$ example_ble AT+F=12
recv from BLE:
+ACK:F,12,1,OK
```

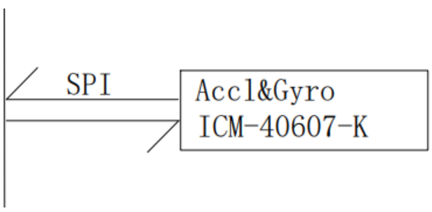
Level change events will be monitored on the PG4 pin

```
$ gpiomon 6 4
event:   RISING EDGE offset: 4 timestamp: [     15970.085069234]
event: FALLING EDGE offset: 4 timestamp: [     15970.085634968]
```

The BLE module can serve as a wake-up source for system sleep, as detailed in the "System Sleep" section.

4.5. G-sensor

Sensor model: ICM-40607-K, connected through SPI bus. The system provides IIO driver and device node /sys/bus/iio/devices/iio: device2.



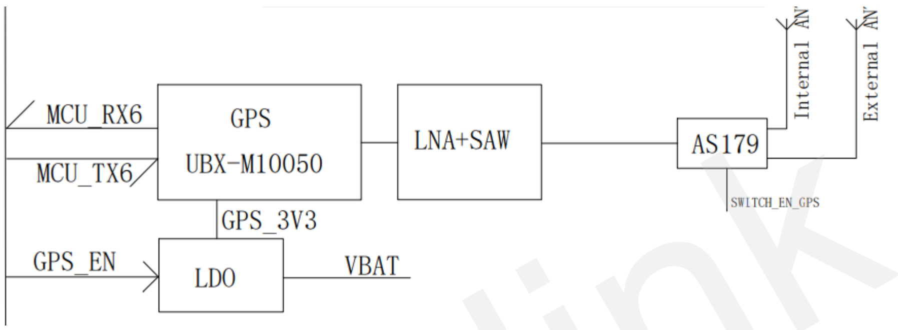
The hardware provides a sensor power supply enable pin. For current GV850, after power on, the output is of high level by default, that is, the G-sensor is turned on by default.

```
$ gpioset 6 0=1
```

Use the provided example_gsensor tool for testing, as detailed in the "Example of Codes" section.

4.6. GPS

Module model: UBX_ M10050, connected through UART. It uses USART6, which corresponds to /dev/ttySTM6. It supports u-blox and NMEA protocols.



The reference testing commands are as follows:

Set the baudrate (38400 by default for M10050),

```
$ stty -F /dev/ttySTM6 ispeed 38400 ospeed 38400 cs8
```

PD13 power supply enable output high level

```
$ gpioset 3 13=1
```

Receive the NMEA data sent by the GPS module

```
$ cat /dev/ttySTM6
```

```
$GNRMC,041722.00,A,2234.41319,N,11356.88217,E,0.002,,050923,,,D,V*11
$GNVTG,,T,,M,0.002,N,0.005,K,D*3F
$GNGGA,041722.00,2234.41319,N,11356.88217,E,2,12,0.52,111.9,M,-2.7,M,,*5A
$GNGSA,A,3,11,15,24,20,23,29,05,13,18,,,,0.97,0.52,0.82,1*06
$GNGSA,A,3,09,36,10,34,05,11,,,,,,,0.97,0.52,0.82,3*0F
$GNGSA,A,3,07,13,28,02,06,59,16,40,27,09,30,20,0.97,0.52,0.82,4*0E
$GNGSA,A,3,,,,,,,,,,,,,0.97,0.52,0.82,5*06
(...)
```

Additionally, NMEA data can be forwarded to RS232_<N> serial port, and then open RS232 through the u-center tool to more intuitively parse NMEA data. The following example is to forward NMEA data to RS232_ 2 serial ports.

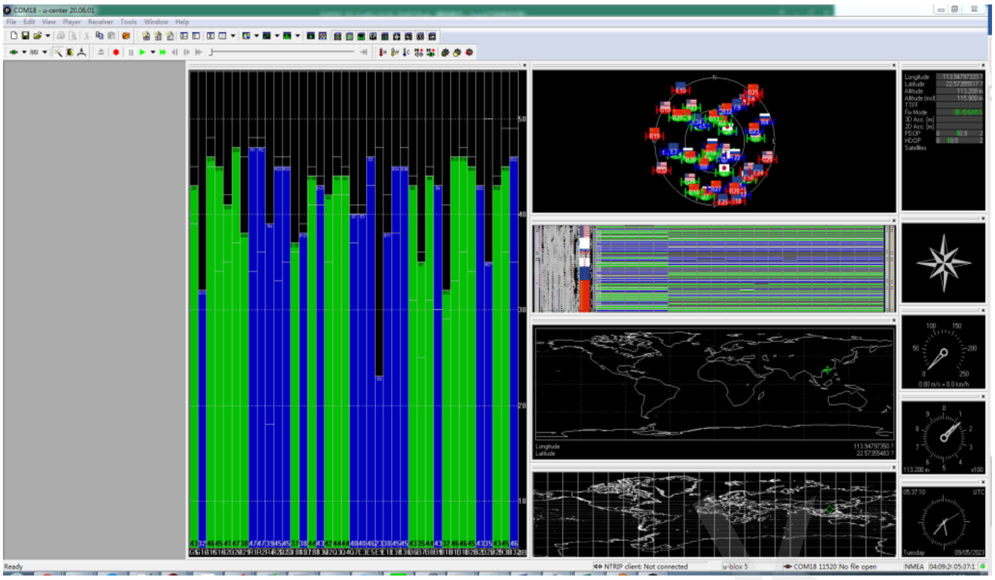
Set RS232_2 baud rate to same 38400:

```
$ stty -F /dev/ttySTM2 ispeed 38400 ospeed 38400 cs8
```

Forward the data to RS232_2:

```
$ cat /dev/ttySTM6 > /dev/ttySTM2
```

Then select the correct COM port and baud rate in the u-center tool to start receiving and parsing data.



Linux system can also provide parsing and control tools that support multi-protocol such as NMEA and u-blox through the integration of gpsd software. For more information on how to use the tools, please refer to the following website:

- https://gpsd.io/

cgps and gpsmon tools can instantly parse and display positioning data. cgps receives and parses JSON data containing positioning data information provided by gpsd services. And gpsmon directly parses and displays the raw data of the GPS module. Depending on the protocol supported by the module, choose to use u-blox or NMEA protocol accordingly.

```
$ cgps
```

```
                                                        Seen 37/Used 28
 Time          2023-09-21T03:01:38.000Z (18)  GNSS   PRN  Elev   Azim   SNR Use
 Latitude           22.57354400 N             GP  5    5  41.0   38.0  43.0  Y
 Longitude         113.94796870 E             GP 11   11  28.0  124.0  30.0  Y
 Alt (HAE, MSL)      346.220,     355.121 ft  GP 13   13  57.0   32.0  45.0  Y
 Speed               0.01 mph                 GP 15   15  71.0  292.0  46.0  Y
 Track (true, var):      0.0,  -3.0      deg  GP 18   18  22.0  323.0  39.0  Y
 Climb              -8.27 ft/min              GP 20   20  25.0   71.0  41.0  Y
 Status           3D DGPS FIX (7 secs)        GP 23   23  12.0  289.0  37.0  Y
 Long Err  (XDOP, EPX)  0.38, +/-  4.7 ft     GP 24   24  30.0  173.0  40.0  Y
 Lat Err   (YDOP, EPY)  0.35, +/-  4.4 ft     GP 29   29  47.0  258.0  44.0  Y
 Alt Err   (VDOP, EPV)  0.82, +/-  5.2 ft     GA  3  303  78.0  330.0  45.0  Y
 2D Err    (HDOP, CEP): 0.50, +/-  3.1 ft     GA  8  308  38.0  235.0  41.0  Y
 3D Err    (PDOP, SEP): 0.96, +/- 15.0 ft     GA 34  334  66.0  106.0  46.0  Y
 Time Err  (TDOP):      0.56                  GA 36  336  17.0  132.0  39.0  Y
 Geo Err   (GDOP):      1.11                  BD  1  401  48.0  121.0  42.0  Y
 Speed Err (EPS)       +/-  0.2 mph           BD  3  403  65.0  190.0  44.0  Y
 Track Err (EPD)        n/a                   BD  6  406  44.0  180.0  40.0  Y
 Time offset           -748330839.226498000   BD  7  407  22.0  197.0  36.0  Y
sGrid Square            OL62xn37              BD  8  408  48.0    6.0  41.0  Y
-More...                                      More...
e,"gnssid":2,"svid":34,"health":1},{"PRN":336,"el":17.0,"az":132.0,"ss":39.0,"us
ed":true,"gnssid":2,"svid":36,"health":1},{"PRN":401,"el":48.0,"az":121.0,"ss":4
```

Queclink Confidential

$ gpsmon

The results of parsing data using the u-blox protocol:

```
/dev/ttySTM6                  u-blox>
Ch PRN  Az  El S/N Flag U    ECEF Pos:
 0   5  37  41  47 191f Y    ECEF Vel:
 1  11 124  29  32 091f Y
 2  13  33  58  47 091f Y    LTP Pos:
 3  15 290  71  48 091f Y    LTP Vel:
 4  18 323  21  39 091f Y
 5  20  71  25  41 091f Y    Time:
 6  23 288  11  39 091f Y    Time GPS:                     Day:
 7  24 173  30  42 091f Y
 8  29 260  47  46 091f Y    Est Pos Err       m Est Vel Err       m/s
 9 127 257  20  40 1a17      PRNs: ## PDOP: xx.x Fix 0x.. Flags 0x..
10 128 237  46  44 1a17                          NAV_SOL
11 129 149  60   0 0701
12 137 149  60  43 1a07      DOP [H]  0.5 [V]  0.8 [P]  1.0 [T]  0.6 [G]  1.1
13 213 327  78  45 091f Y                        NAV_DOP
14 218 234  38  40 091f Y
15 223 321   2   0 1210      TOFF: > 1 day              PPS:      N/A
       NAV-SAT
(26) b56201041200c05a3e156f006000380052003100220023005З3ad
(24) b56201201000c05a3e150c86f8ffe80812070700000376b
```

$ gpsmon –n

The results of parsing data using the NMEA protocol:

```
/dev/ttySTM6                   NMEA0183>
Time: 2023-09-21T03:02:29.000Z   Lat: 22 34.413000' N   Lon: 113 56.877900' E
                                 Cooked TPV
 GPZDA GPGGA GPRMC GPSGA GPGBS GPGSV
                                 Sentences
 SVID  PRN  Az El SN HU Time:        030229.00      Time:      030229.00
GP   5   5  38 41 44  Y Latitude:     2234.4130 N   Latitude:  2234.4130
GP  13  13  32 57 45  Y Longitude:   11356.8779 E   Longitude: 11356.8779
GP  15  15 293 71 46  Y Speed:       0.0078         Altitude:  106.85
GP  18  18 323 22 40  Y Course:      0.000          Quality:   2   Sats: 32
GP  20  20  72 24 40  Y Status:      A        FAA:  HDOP:      0.50
GP  23  23 289 12 39  Y MagVar:      -3.0 W         Geoid:     -2.98
GP  24  24 173 31 40  Y              RMC                       GGA
GP  29  29 258 47 44  Y
GP   0   0 149 60 40  N Mode: A3 Sats: 5 13 15 +    UTC:          RMS:
GP  11  11 125 28 22  N DOP H=0.5   V=0.8   P=1.0   MAJ:          MIN:
GP  30  30  44  5  0  N TOFF: > 1 day               ORI:          LAT:
SB127   40 257 20 39  N PPS: N/A                    LON:          ALT:
    v      GSV              GSA + PPS                     GST
(76) $GPGSV,10,10,40,194,12,151,38,195,57,141,45,196,65,049,44,199,60,149,40*79
```

Check u-blox version:

$ ubxtool -p MON-VER

UBX-MON-VER:

swVersion ROM SPG 5.10 (7b202e)

hwVersion 000A0000

extension FWVER=SPG 5.10

extension PROTVER=34.10

extension GPS;GLO;GAL;BDS

extension SBAS;QZSS

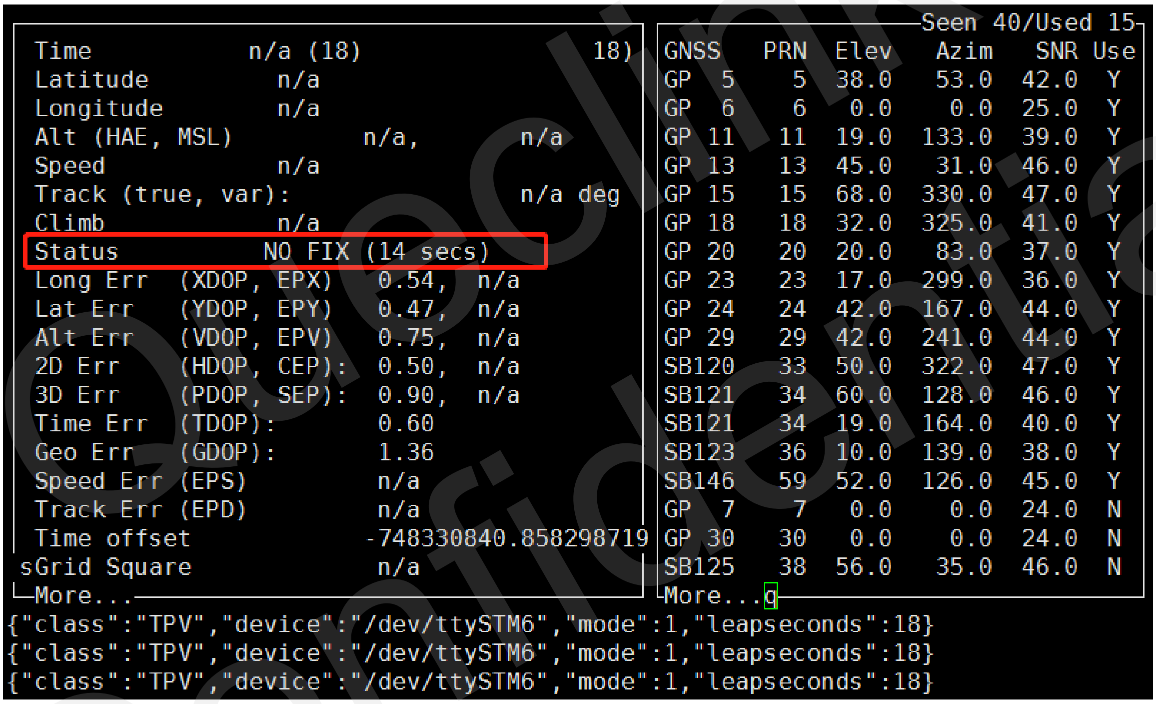
WARNING: protVer is 10.00, should be 34.10.   Hint: use option "-P 34.10"

Queclink Confidential

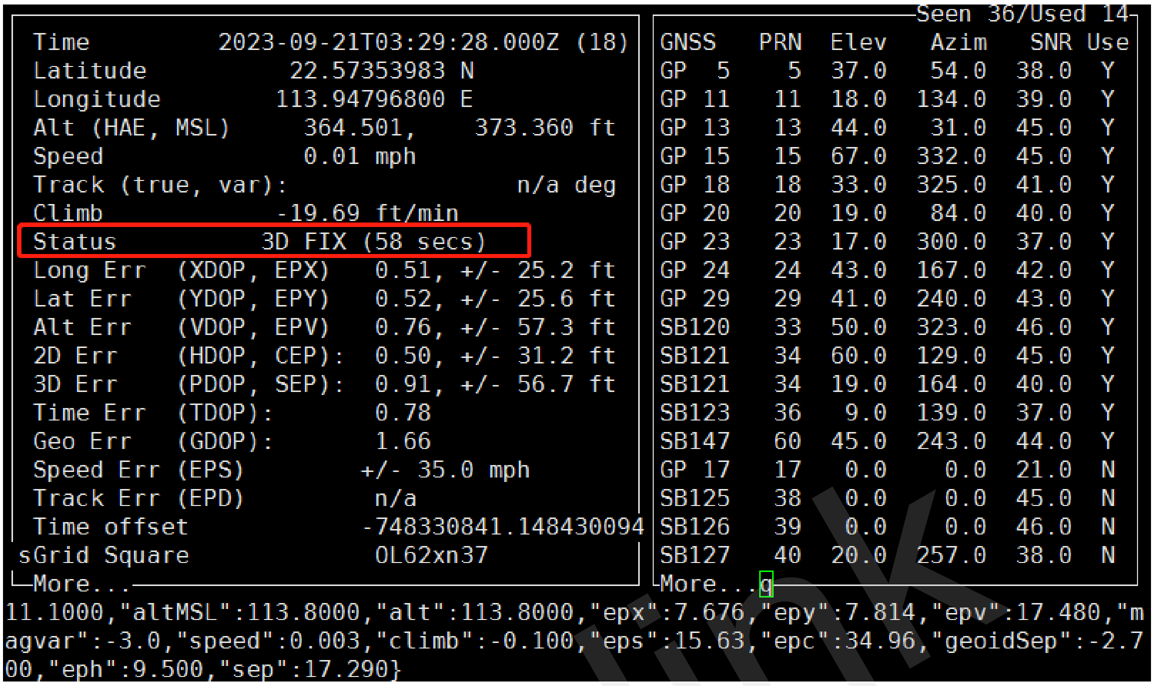
```
UBX-NAV-PVT:
   iTOW 357228000 time 2023/9/21 03:13:30 valid x37
   tAcc 24 nano -443396 fixType 3 flags x3 flags2 xea
   numSV 32 lon 1139479540 lat 225735412 height 110133
   hMSL 112846 hAcc 580 vAcc 1212
   velN -1 velE 2 velD 17 gSpeed 2 headMot 0
   sAcc 112 headAcc 17333086 pDOP 103 reserved1 0 16476 12118
   headVeh 3102272 magDec 0 magAcc 0
   (...)
```

You can use the following commands to perform cold start and calculate the time it takes from no positioning to positioning by the status change of the cgps monitoring tool:

```
$ ubxtool -p COLDBOOT -P 34.10
```



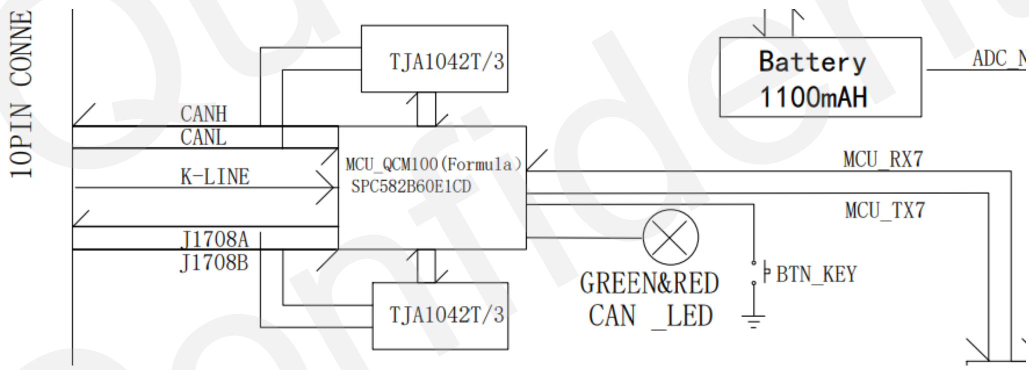
Queclink Confidential



```
 Time          2023-09-21T03:29:28.000Z (18) | GNSS   PRN  Elev   Azim   SNR Use
 Latitude           22.57353983 N            | GP  5    5  37.0   54.0  38.0  Y
 Longitude         113.94796800 E            | GP 11   11  18.0  134.0  39.0  Y
 Alt (HAE, MSL)      364.501,    373.360 ft  | GP 13   13  44.0   31.0  45.0  Y
 Speed               0.01 mph                | GP 15   15  67.0  332.0  45.0  Y
 Track (true, var):                n/a deg   | GP 18   18  33.0  325.0  41.0  Y
 Climb             -19.69 ft/min             | GP 20   20  19.0   84.0  40.0  Y
 Status          3D FIX (58 secs)            | GP 23   23  17.0  300.0  37.0  Y
 Long Err  (XDOP, EPX)   0.51, +/- 25.2 ft   | GP 24   24  43.0  167.0  42.0  Y
 Lat Err   (YDOP, EPY)   0.52, +/- 25.6 ft   | GP 29   29  41.0  240.0  43.0  Y
 Alt Err   (VDOP, EPV)   0.76, +/- 57.3 ft   | SB120   33  50.0  323.0  46.0  Y
 2D Err    (HDOP, CEP):  0.50, +/- 31.2 ft   | SB121   34  60.0  129.0  45.0  Y
 3D Err    (PDOP, SEP):  0.91, +/- 56.7 ft   | SB121   34  19.0  164.0  40.0  Y
 Time Err  (TDOP):       0.78                | SB123   36   9.0  139.0  37.0  Y
 Geo Err   (GDOP):       1.66                | SB147   60  45.0  243.0  44.0  Y
 Speed Err (EPS)        +/- 35.0 mph         | GP 17   17   0.0    0.0  21.0  N
 Track Err (EPD)         n/a                 | SB125   38   0.0    0.0  45.0  N
 Time offset            -748330841.148430094 | SB126   39   0.0    0.0  46.0  N
sGrid Square             OL62xn37            | SB127   40  20.0  257.0  38.0  N
More...                                      | More...q          Seen 36/Used 14
11.1000,"altMSL":113.8000,"alt":113.8000,"epx":7.676,"epy":7.814,"epv":17.480,"m
agvar":-3.0,"speed":0.003,"climb":-0.100,"eps":15.63,"epc":34.96,"geoidSep":-2.7
00,"eph":9.500,"sep":17.290}
```

4.7. CAN

Module model: SPC582B60E1, connected through UART. It uses USART6, which corresponds to /dev/ttySTM7



Set the baud rate (default) to 115200, and because the module serial port data is binary, the parameter raw needs to be used when using the stty tool to set it. Otherwise, the default tty attribute may overwrite the read data, such as the enabled icrnl attribute by default, which will overwrite 0x0D with 0x0A.

```
$ stty -F /dev/ttySTM7 ispeed 115200 ospeed 115200 cs8 raw
```

PG3 CAN MCU Power supply output enable：

```
$ gpioset 6 3=1
```

PA4 5V voltage increase enable：

```
$ gpioset 0 4=1
```

On Linux system, the read and written binary data can be edited by using the hexedit tool, and then read and write by using the dd tool.

For example, write the binary command to be sent into the file out:

Queclink Confidential

```
$ touch out
$ hexedit out
```

```
00000000 F5 B3 10 01 3B F6                          ....;.
00000010
00000020
00000030
00000040
00000050
00000060
```

Start reading in advance (at the background) and write the read data to the in file,

```
$ dd if=/dev/ttySTM7 of=in &
```

Send the out file,

```
$ dd of=/dev/ttySTM7 if=out
```

Use the hexdump tool to display the read binary data.

```
$ hexdump -C in
```

```
root@Queclink-GV850:/tmp# hexdump -C in
00000000  f5 b4 14 01 49 30 08 0d  a8 f6             |....I0....|
0000000a
root@Queclink-GV850:/tmp#
```

The module command/protocol description is detailed in the document "[22-12-12] CAN-Logistic v3 protocol XON-XOFF". The module provides three configurable GPIO outputs, where OUT2 is connected to PG1 of the MPU and can notify the MPU of events. The testing method is as follows:

Query current PG1 status:

```
$ gpioget 6 1
1
```

Use example_external_can tool to make OUT2 output 0,

```
$ example_external_can 0x402
STEP
Read version, write len=6:
F5 B3 10 01 3B F6
read len=10:
F5 B4 14 01 49 30 08 0D A8 F6

STEP
GPIO output1~3 disactivated, write len=9:
F5 B3 43 02 00 00 20 E7 F6
read len=9:
F5 B4 43 02 00 6E 00 98 F6
```

Query PG1 status again and it is updated,

Queclink Confidential

```
$ gpioget 6 1
0
```

make OUT2 output 1,

```
$ example_external_can 0x802
STEP
Read version, write len=6:
F5 B3 10 01 3B F6
read len=10:
F5 B4 14 01 49 30 08 0D A8 F6

STEP
GPIO output2 activated, write len=9:
F5 B3 43 02 40 00 20 A7 F6
read len=9:
F5 B4 43 02 00 6E 00 98 F6
```

Query current PG1 status, and it goes back to the beginning status,

```
$ gpioget 6 1
1
```

gpiomon can also be used to monitor PG1,

```
$ gpiomon 6 1
event:  RISING EDGE offset: 1 timestamp: [     2915.299024187]
event: FALLING EDGE offset: 1 timestamp: [     2915.317364562]
```

The CAN module can serve as a wake-up source for system sleep. On hardware, OUT2 is connected to PG1 of MPU as the wake-up source. The OUT2 function is configurable, with the default function being 'vehicle's buses active',

6.6.9. Setting outputs functions

CAN-Logistic has three binary outputs (bistable), which can provide various signals related to vehicle state. Default outputs functions are:

- ignition on for OUT1
- vehicle's buses active for OUT2
- notification about events for OUT3

Outputs may be positive (high level voltage when active), or negative (shorted to ground when active) - check hardware information.

The module will enter sleep on its own and pull PG1 up. When the module is awakened, pressing the CAN sync button will pull PG1 down,

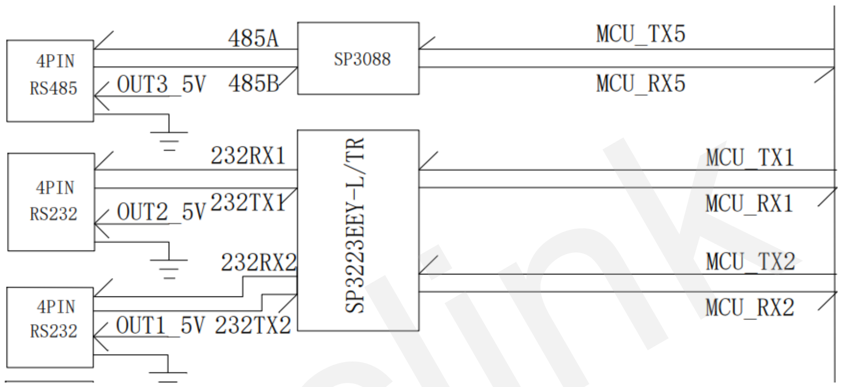
```
$ gpiomon 6 1
event: FALLING EDGE offset: 1 timestamp: [   156339.620835030]
```

Queclink Confidential

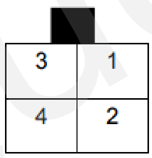
4.8. RS232/RS485

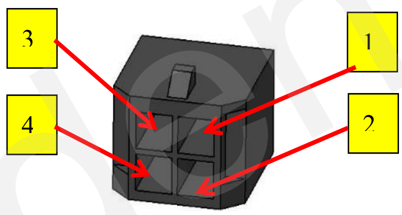
There are 2 RS232 and 1 RS485.

4-pin	Hardware	Device	Description
RS485	USART5	/dev/ttySTM5	/
RS232_1	USART1	/dev/ttySTM1	/
RS232_2	USART2	/dev/ttySTM2	/



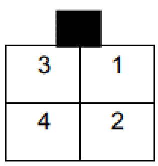
The front view of the 4-pin RS485 connector is as follows:

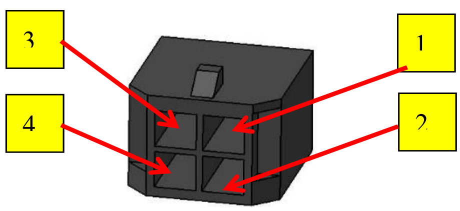




Pin	Pin Name	Cable Color	Description	Device Nodes	Remarks
1	GND	Black	External Accessory Ground	/	/
2	DC5V_3	Red	External Accessory Power 250mA Max	/	/
3	485B	Orange white	RS485B	/	/
4	485A	Orange black	RS485A	/	/

The front view of the 4-pin RS232 connectors is as follows:



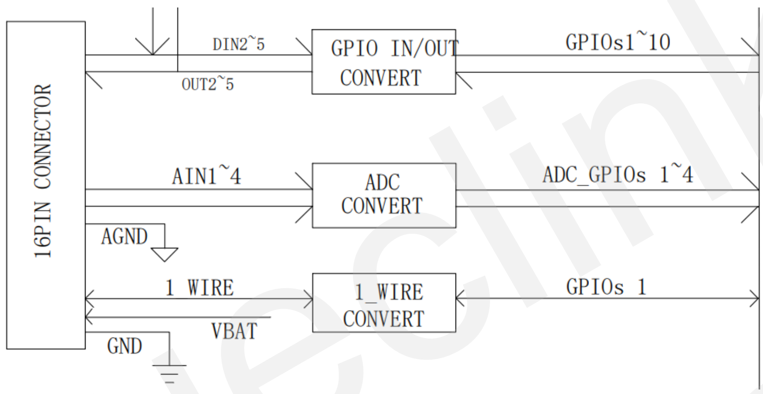


Queclink Confidential

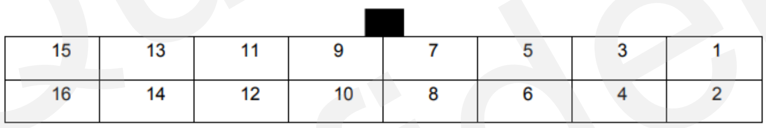
Pin	Pin Name	Cable Color	Description	Device Nodes	Remarks
1	GND	Black	External Accessory Ground	/	/
2	DC5V_1	Red	External Accessory Power 250mA Max	/	/
3	TX232_1	Gray black	UART TXD1 RS232	/	/
4	RX232_1	Gray white	UART RXD1 RS232	/	/

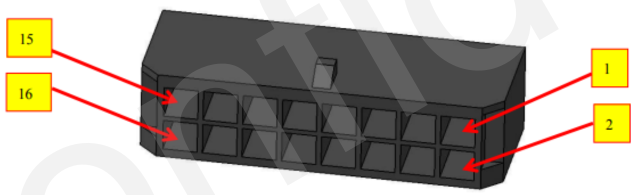
4.9. GPIO&ADC&1-WIRE

There are 10 GPIO, 1 1-wire bus and 4 ADC inputs.



The front view of the 16-pin connector is as follows:





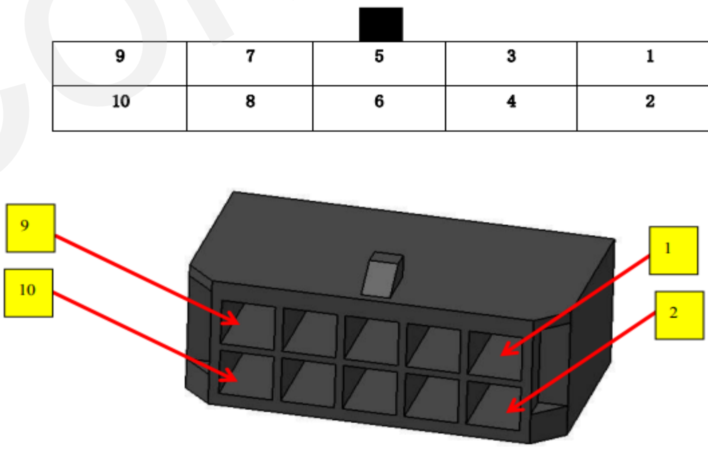
Descriptions of IOs and ADCs are as follows:

Pin	Pin Name	Cable Color	Description	Device Nodes	Remarks
1	AIN1	Brown/white	Analog Input1 0~32V	/sys/bus/iio/devices/iio:device1/in_voltage_scale /sys/bus/iio/devices/iio:device1/in_voltage10_raw	Volt=scale*raw*(18+200)/18 Unit: mV
2	DIN2	Orange/black	Negative trigger input2	gpiochip2 9	$ gpioget gpiochip2 9
3	AIN2	Red/brown	Analog Input2 0~32V	/sys/bus/iio/devices/iio:device1/in_voltage_scale /sys/bus/iio/devices/iio:d	Volt=scale*raw*(18+200)/18 Unit: mV

Queclink Confidential

				evice1/in_voltage4_raw	
4	DIN3	Blue	Negative trigger input3	gpiochip2 10	$ gpioget gpiochip2 10
5	AIN3	White/black	Analog Input3 0~32V	/sys/bus/iio/devices/iio:device0/in_voltage_scale /sys/bus/iio/devices/iio:device0/in_voltage2_raw	Volt=scale*raw*(18+200)/18 Unit: mV
6	DIN4	Black/brown	Negative trigger input4	gpiochip2 11	$ gpioget gpiochip2 11
7	AIN4	Gray/black	Analog Input4 0~32V	/sys/bus/iio/devices/iio:device1/in_voltage_scale /sys/bus/iio/devices/iio:device1/in_voltage0_raw	Volt=scale*raw*(18+200)/18 Unit: mV
8	DIN5	Pink	Negative trigger input5	gpiochip2 12	$ gpioget gpiochip2 12
9	OUT3	Brown	Open drain output3	gpiochip8 0	$ gpioset gpiochip8 0=value
10	OUT5	Orange	Open drain output5	gpiochip8 3	$ gpioset gpiochip8 3=value
11	OUT2	Yellow	Open drain output2	gpiochip6 15	$ gpioset gpiochip6 15=value
12	OUT4	White	Open drain output4	gpiochip8 2	$ gpioset gpiochip8 2=value
13	1W_DATA	Green	1-WIRE data	/	/
14	GND	Black	Ground	/	/
15	VDD_1WIRE	Red white	Power for 1-wire devices 3.3V	/	/
16	AGND	Black gray	Analog Ground	/	/

The front view of the 10-pin connector is as follows:



Queclink Confidential

Descriptions of IOs and ADCs are as follows:

Pin	Pin Name	Cable Color	Description	Device Nodes	Remarks
1	DCIN	Red	DC Power 8-32V	/	/
2	GND	Black	Ground	/	/
3	IGN	White	Positive trigger input	gpiochip0 6	$ gpioget gpiochip0 6
4	DIN1	Orange	Negative trigger input1	gpiochip2 8	$ gpioget gpiochip2 8
5	K-LINE	Pink	ISO K Line	/	/
6	OUT1	Yellow	Open drain output1 with latch	gpiochip0 3	$ gpioset gpiochip0 3=value
7	CAN1L	Brown black	CAN Bus CAN1L	/	/
8	CAN1H	Brown white	CAN Bus CAN1H	/	/
9	CAN2L	Blue	CAN Bus CAN2L	/	/
10	CAN2H	Brown	CAN Bus CAN2H	/	/

The STM32MP133 platform can use the gpio tools tool to print GPIO group information.

```
$ gpiodetect
gpiochip0 [GPIOA] (16 lines)
gpiochip1 [GPIOB] (16 lines)
gpiochip2 [GPIOC] (16 lines)
gpiochip3 [GPIOD] (16 lines)
gpiochip4 [GPIOE] (16 lines)
gpiochip5 [GPIOF] (16 lines)
gpiochip6 [GPIOG] (16 lines)
gpiochip7 [GPIOH] (15 lines)
gpiochip8 [GPIOI] (8 lines)
```

```
# gpioinfo
gpiochip0 - 16 lines:
        line   0:      "PA0"      kernel   input  active-high [used]
        line   1:      "PA1"      unused   input  active-high
        line   2:      "PA2"      kernel   input  active-high [used]
        line   3:      "PA3"      unused   input  active-high
        line   4:      "PA4"      unused   input  active-high
        line   5:      "PA5"      kernel   input  active-high [used]
        (...)
```

4.10. Power&Battery

Main power function and interface description are as follows:

Function	Device Nodes	Remarks
Voltage detection	/sys/bus/iio/devices/iio:device1/in_voltage_scale /sys/bus/iio/devices/iio:device1/in_voltage2_raw	Volt=scale*raw*(82+1000)/82 Unit: mV

Backup battery power function and interface description are as follows:

Function	Device Nodes	Remarks
Voltage detection	/sys/bus/iio/devices/iio:device1/in_voltage_scale /sys/bus/iio/devices/iio:device1/in_voltage1_raw	Volt=scale*raw*(200+200)/200 Unit: mV

Power supply On	gpiochip5 12	On
		$ gpioset gpiochip5 12=1
		Off
		$ gpioset gpiochip5 12=0

Power supply Off		

Charging Start	gpiochip0 11	Start
		$ gpioset gpiochip0 11=1
		Stop
		$ gpioset gpiochip0 11=0

Charging Stop		
Charging Status	gpiochip6 12	$ gpioget gpiochip6 12 0, Charging 1, Not Charging

Charging IC On	gpiochip0 13	On
		$ gpioset gpiochip0 13=1
		Off
		$ gpioset gpiochip0 13=0
		The power supply of the ammeter IC is associated with the power supply input of the battery charging management IC. It is necessary to enable it first (backup battery on, charging ammeter IC on), then enable charging (backup battery charging starts), and then detect the charging current

Charging IC Off		
Charging Current	/sys/bus/iio/devices/iio:device1/in_voltage_scale /sys/bus/iio/devices/iio:device1/in_voltage16_raw	Current=scale*raw Unit: mA

Battery Temperature Detection On	gpiochip6 13	On
		$ gpioset gpiochip6 13=1
		Off
		$ gpioset gpiochip6 13=0

Battery Temperature Detection Off		
Battery	/sys/bus/iio/devices/iio:device1/in_voltage_scale	Volt=scale*raw*(10+10)/10

Queclink Confidential

Temperature	/sys/bus/iio/devices/iio:device1/in_voltage15_raw	Unit: mV

4.11. LED

There are 3 LEDs, which are controlled by GPIO.

Pin Name	LED Name	Description

PA8	GPS signal LED	On, PA8 output high
		$ gpioset 0 8=1
		Off, PA8 output low
		$ gpioset 0 8=0

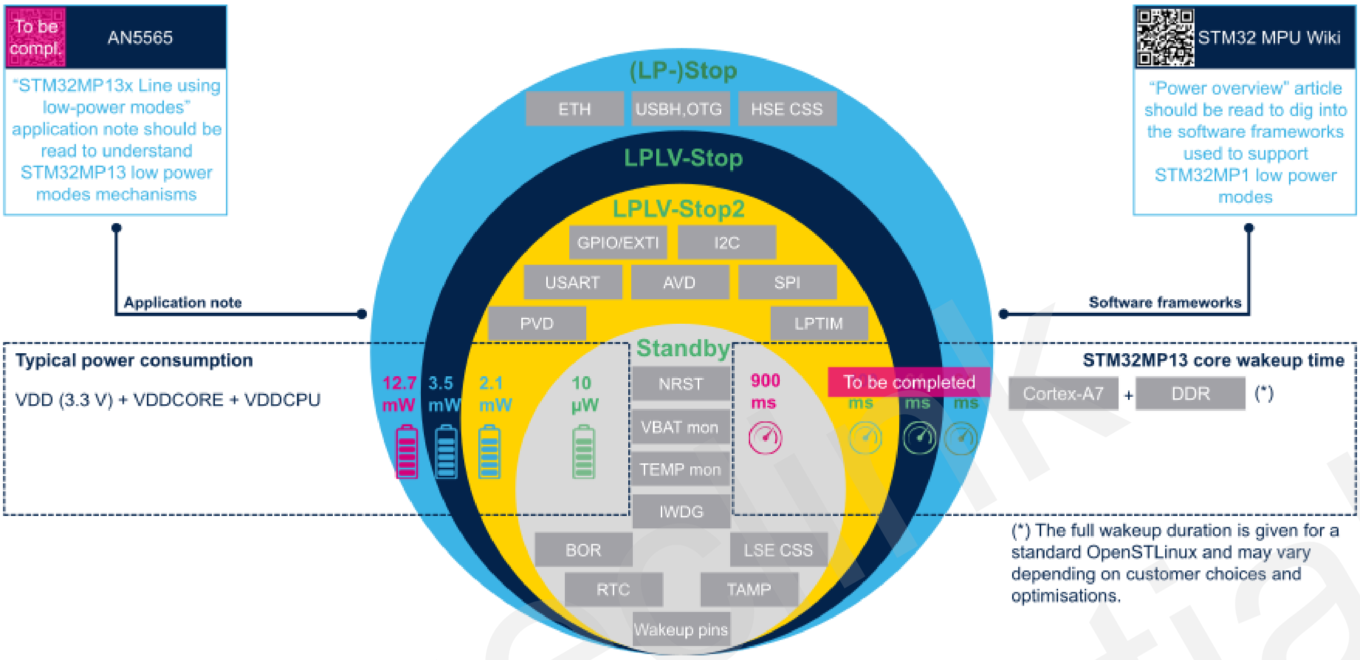
PE6	Network signal LED	On, PE6 output high
		$ gpioset 4 6=1
		Off, PE6 output low
		$ gpioset 4 6=0

PG11	Power supply status LED	On, PG11 output high
		$ gpioset 6 11=1
		Off, PG11 output low
		$ gpioset 6 11=0

Queclink Confidential

5. System Sleep

This section introduces the low-power design of the stm32mp133 platform and the control methods for entering low-power. MPU provides multiple energy consumption operation modes,



The wake-up sources supported in each mode are different, as shown in the following table,

Platform mode	Available wakeup sources
Stop	BOR, PVD, AVD, Vbat mon, Temp mon, HSE CSS, LSE CSS, RTC, TAMP, USBH, OTG, ETH, USART, I2C, SPI, DTS, LPTIM, IWDG, GPIO, Wakeup pins (from PWR)
LPLV-Stop LPLV-Stop2	BOR, PVD, AVD, Vbat mon, Temp mon, LSE CSS, RTC, TAMP, USART, I2C, SPI, DTS, LPTIM, IWDG, GPIO, Wakeup pins (from PWR)
Standby	BOR, Vbat mon, Temp mon, LSE CSS, RTC, TAMP, IWDG, Wakeup pins (from PWR)

That is to say, the low-power mode MPU can enter depends on the wake-up source required by the application scenario,

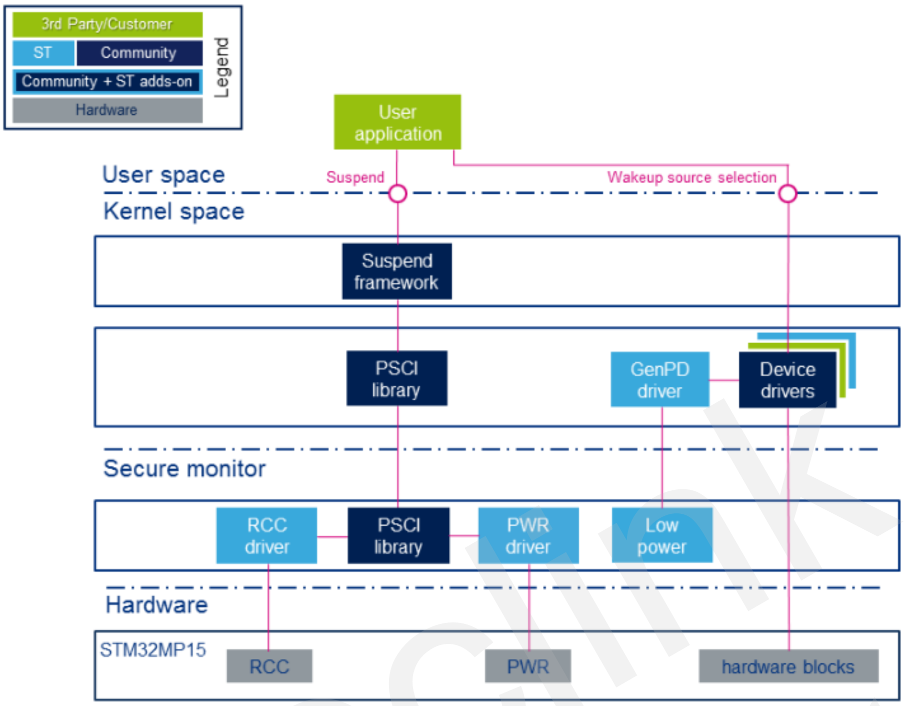
Queclink Confidential

Table 9. Deepest power mode per wake-up source group and equivalence between Linux and STM32MP13x device system power modes

Wake-up source	Linux command	STM32MP13x device system deepest power mode	System DDR	Linux kernel state	Power consuming	Wake-up time	Comment/Application guideline
Group 1: USB, CEC, ETH	"mem"	Stop or LP-Stop	SR (VTT off)	"Suspend-to-ram"	Medium	Medium	LP-Stop: driving external PWR_LP/ PWR_ON permits designing the custom strategy for the external regulator. Typical application is to switch off DDR3 termination supply (VTT) (most likely not needed in 16-bit DDR design)
Group 2: PVD, AVD, DTS, USART, I²C, SPI, LPTIM, GPIOs	"mem"	LPLV-Stop or LPLV-Stop2	SR (VTT off)	"Suspend-to-ram"	Low	Medium	LPLV-Stop(2): save power thanks to the power retention. Suitable for applications with aggressive power constraints and tolerant with limitations of wake-up source (refer to Table 4. Low-power mode wake-up capabilities of the system)
Group 3: BOR, Vbat mon, Temp mon, LSE CSS, RTC, TAMP, wake-up pins	"mem"	Standby	SR	"Suspend-to-ram"	Low	Medium	Standby saves more power at the expense of wake-up time
	"shutdown"	Off/VBAT	Off	Shutdown	Very low	High	-

According to the GV850 specifications and application scenarios, it is required to realize modes LPLV-STOP/LPLV-STOP2 and Off/VBAT. OpenSTLinux implements a power management mechanism, as shown in the following figure,

Queclink Confidential



Only by using the provided Linux sysfs interface, the configuration and enabling/disabling of wake-up sources and initiating of state/mode switchover request can be done. By calling the PWR driver to control the hardware PWR, adjust the VDDCORE and VDDCPU voltages according to the following table. After both voltages meet the conditions, the MPU as a whole can enter the corresponding energy consumption state.

Due to differences in power management hardware between GV850 and the official demo board, GV850 uses separate components instead of power management IC (PMIC), and GPIO is used for PWR control instead of I2C interface. Therefore, GPIO needs to be adapted and adopted in the PWR driver.

Table 8. **STPMIC1x (LP mode) programming: LP-Stop LPLV-Stop and Standby mode**

Supply name	Control register (LP mode) /@	LP-Stop	LPLV-Stop	LPLV-Stop2	Standby with DDR SR	Standby w/o DDR SR
V_{DDCORE}	BUCK4/0x33	0x69 (1.25 V)	0x33 (0.9 V)	0x33 (0.9 V)	0x30 (off)	
V_{DDCPU}	BUCK1/0x30	0x69 (1.25 V)	0x33 (0.9 V)	0x30 (off)		
V_{DD_DDR}	BUCK2/0x31	0x79 (1.35 V)				0x7A (off)
V_{DD}	BUCK3/0x32	0xD9 (3.3 V)				
V_{REF_DDR}	VREFDDR/0x34	0x1				0x0 (off)
V_{DDA}	LDO1/0x35	0x51 (2.9 V)			0x50 (off)	
V_{DD_USB}	LDO4/0x38	0x1 (3.3 V)			0x0 (off)	
V_{DD_SD}	LDO5/0x39	0x51 (2.9 V)			0x50 (off)	

Queclink Confidential

Wake-up source table,

Name	Pin	Location
RTC		Inside MPU
UART		Interface
LTE	GPIO PD3	External module
USB	GPIO PD7	Interface
CAN	GPIO PG1	External module
BLE	GPIO PG4	External module
G-sensor		External module

5.1. RTC Wake-up

It is enabled by default. By using the rtcwake tool, scheduled wake-up can be completed. The usage method is as follows:

Check the current system time,

```
$ date
Wed Sep 27 14:36:57 UTC 2023
```

Initiate sleep and wake up at 14:39,

```
$ rtcwake -t `date -d 14:39 +%s` -m mem -d /dev/rtc0
wakeup from "mem" at Wed Sep 27 14:38:58 2023
[  825.648590] PM: suspend entry (deep)
[  825.651151] Filesystems sync: 0.000 seconds
[  825.662977] Freezing user space processes ... (elapsed 0.001 seconds) done.
[  825.670398] OOM killer disabled.
[  825.673390] Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.
[  825.680988] printk: Suspending console(s) (use no_console_suspend to debug)
```

Or directly specify the sleep interval,

```
$ rtcwake -s 60 -m mem -d /dev/rtc0
wakeup from "mem" at Sat Jan  1 22:54:38 2000
[   27.544938] PM: suspend entry (deep)
[   27.547565] Filesystems sync: 0.000 seconds
[   27.559313] Freezing user space processes ... (elapsed 0.001 seconds) done.
[   27.566787] OOM killer disabled.
[   27.569771] Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.
[   27.577385] printk: Suspending console(s) (use no_console_suspend to debug)
```

Will wake up after 60 seconds of sleep by itself and return to the system command prompt.

```
[   27.584948] inv-mpu-iio-spi spi0.0: icm42600 suspend
[   27.588715] dwc2 49000000.usb-otg: suspending usb gadget g_ether
[   27.593181] Disabling non-boot CPUs ...
[   27.597019] dwc2 49000000.usb-otg: resuming usb gadget g_ether
[   27.602441] nand: SDR timing mode 4 not acknowledged by the NAND chip
```

```
[    27.604035] inv-mpu-iio-spi spi0.0: icm42600 resume
[    27.635996] OOM killer enabled.
[    27.639113] Restarting tasks ... done.
[    27.657140] PM: suspend exit
root@Queclink-GV850:~#
```

5.2. UART Wake-up

It is disabled by default. Taking the system console UART device ttySTM0 as an example to show the enabling method,

Check the default value,

```
$ cat /sys/devices/platform/soc/40010000.serial/power/wakeup
disabled
$ cat /sys/devices/platform/soc/40010000.serial/tty/ttySTM0/power/wakeup
disabled
```

Modify the wake-up source to enable state,

```
$ echo enabled > /sys/devices/platform/soc/40010000.serial/tty/ttySTM0/power/wakeup
$ echo enabled > /sys/devices/platform/soc/40010000.serial/power/wakeup
```

Initiate a sleep request,

```
$ echo mem > /sys/power/state
[   192.680917] PM: suspend entry (deep)
[   192.695818] Filesystems sync: 0.012 seconds
[   192.699298] Freezing user space processes ... (elapsed 0.001 seconds) done.
[   192.706747] OOM killer disabled.
[   192.709813] Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.
[   192.717339] printk: Suspending console(s) (use no_console_suspend to debug)
```

During sleep, if there are no other wake-up sources, it will wake up when UART receives data and return to the system command line login.

```
[   192.725008] inv-mpu-iio-spi spi0.0: icm42600 suspend
[   192.728334] dwc2 49000000.usb-otg: suspending usb gadget g_ether
[   192.732793] Disabling non-boot CPUs ...
[   192.736417] dwc2 49000000.usb-otg: resuming usb gadget g_ether
[   192.741961] nand: SDR timing mode 4 not acknowledged by the NAND chip
[   192.743202] inv-mpu-iio-spi spi0.0: icm42600 resume
[   192.775183] OOM killer enabled.
[   192.778435] Restarting tasks ... done.
[   192.783454] PM: suspend exit
^Z
Welcome to Buildroot
Queclink-GV850 login:
```

6. Example of Codes

In order to facilitate developers to familiarize themselves with and use the modules on the device, example source code for some module interfaces is provided for reference.

6.1. example_ble

It demonstrates how to send commands to the BLE module and receive response data. For more information on the module, please refer to the "BLE" section.
The method is as follows, with the main steps being to enable power supply and test command:

```
$ gpioset 4 15=1
```

Use the tool to send the AT+F=1 command to read the BLE firmware version. Since it has just started up, the response data is a startup message. Please ignore it,

```
$ example_ble AT+F=1
recv from BLE:
+ACK:X,99,01.01,64F4193F0C000000
```

Use the same command again, it returns the correct response data, including BLE firmware version information,

```
$ example_ble AT+F=1
recv from BLE:
+ACK:F,1,01.01,OK
```

Query the BOOT APP version of the BLE module,

```
$ example_ble AT+F=17,0
recv from BLE:
+ACK:F,17,0,NABE5_BT_BOOTR00A01V01,OK
$ example_ble AT+F=17,1
recv from BLE:
+ACK:F,17,1,NABE5_BT_R00A02V03,OK
```

6.2. example_formula_can

It demonstrates how to send commands to the CAN module and receive response data. For more information on the module, please refer to the "CAN" section.
The method is as follows, with the main steps being to set baud rate, enable power supply and test command:

```
$ stty -F /dev/ttySTM7 ispeed 115200 ospeed 115200 cs8 raw
$ gpioset 6 3=1
$ gpioset 0 4=1
```

Embedded commands inside the tool, parameters (OR values) can be used to control the sequence of the commands to be executed,

```
$ example_external_can
Usage:
```

```
    example_external_can <testing mask>

Testing mask:
    -Read SN,        --0x01
    -Read version,        --0x02
    -Read boot version,        --0x04
    -Read INPUT_3 voltage,        --0x08
    -Read V_IN voltage,        --0x10
    -Enter develop mode,     --0x20
    -CAN loop test,       --0x40
    -K-Line test,     --0x80
```

Execute Read version command,

```
$ example_external_can 0x02
STEP
Read version, write len=6:
F5 B3 10 01 3B F6
read len=10:
F5 B4 14 01 49 30 08 0D A8 F6
```

Execute Read version and CAN loop test commands,

```
$ example_external_can 0x42
STEP
Read version, write len=6:
F5 B3 10 01 3B F6
read len=10:
F5 B4 14 01 49 30 08 0D A8 F6

STEP
CAN loop test, write len=9:
F5 B3 43 02 00 80 10 77 F6
read len=9:
F5 B4 43 02 00 6E 00 98 F6
```

6.3. example_modem_at

It demonstrates how to send commands to the LTE module and receive response data. For more information on the module, please refer to the "LTE" section.

The method is as follows, with the main steps being to set baud rate, enable power supply, power on the module, turn off command echo and test command:

```
$ stty -F /dev/ttySTM3 ispeed 115200 ospeed 115200 cs8 -icrnl -isig -icanon -echo -echoe
$ gpioset 0 15=1
$ gpioset 5 5=1
$ gpioset 5 5=0
```

Use the tool to send the ATE0 command to turn off echo,

```
$ example_modem_at ATE0
ATE0


OK
```

Send the AT+GMR command to query the firmware version of the LTE module,

```
$ example_modem_at AT+GMR
EG915UEUABR02A05M08


OK
```

6.4. example_gsensor

Demonstrates how to provide a sysfs interface through the driver to complete the initialization, data collection, and command testing of the IMU device:

```
$ example_gsensor
Usage:
        test-sensors-sysfs [-d <device_no>] [-a <rate>] [-g <rate>] [-c]

Options:
      -h, --help
                Show this help and quit.
      -d, --device
                Choose device by numero.
      -a, --accel
                Turn accelerometer on with ODR (Hz).
      -g, --gyro
                Turn gyroscope on with ODR (Hz).
      -c, --convert
                Show data after unit conversion (m/s^2, rad/s)
      -b, --batch
                Set batch timeout in ms.
Version:
      1.1.0
```

For example, the sampling frequency is 100Hz,

```
$ example_gsensor -d 2 -a 100 -g 100
…
Accel body (LSB)   ,    +113,     +13,   +4077,          17478588377202,   176.818,      1.128
```

Queclink Confidential

```
Gyro   body (LSB)   ,      -4,      +5,      +1,        17478588342172,   16.954,      1.163
Accel body (LSB)   ,    +114,     +13,   +4082,        17478598377202,   10.000,      1.338
Gyro   body (LSB)   ,      -3,      +5,      +1,        17478598307142,    9.965,      1.408
Accel body (LSB)   ,    +111,      +8,   +4084,        17478608377202,   10.000,      1.176
Gyro   body (LSB)   ,      -5,      +6,      +1,        17478608272112,    9.965,      1.281
Accel body (LSB)   ,    +111,     +12,   +4094,        17478618377202,   10.000,      1.226
Gyro   body (LSB)   ,      -4,      +4,      +1,        17478618237082,    9.965,      1.366
Accel body (LSB)   ,    +114,     +14,   +4092,        17478628377202,   10.000,      1.109
Gyro   body (LSB)   ,      -4,      +5,      +0,        17478628202052,    9.965,      1.284
```

Sampling results after unit conversion,

```
$ example_gsensor -d 2 -a 100 -g 100 -c
…
Accel body (m/s^2),    +0.270545,    +0.021548,    +9.761160,     17568890862843,  169.503,    1.343
Gyro   body (rad/s),    -0.005326,    +0.005326,    +0.001065,     17568890827813,    9.549,    1.378
Accel body (m/s^2),    +0.270545,    +0.028730,    +9.806650,     17568900862843,   10.000,    1.376
Gyro   body (rad/s),    -0.004261,    +0.005326,    +0.001065,     17568900792783,    9.965,    1.446
Accel body (m/s^2),    +0.268151,    +0.021548,    +9.782708,     17568910862843,   10.000,    1.426
Gyro   body (rad/s),    -0.005326,    +0.005326,    +0.000000,     17568910757753,    9.965,    1.531
Accel body (m/s^2),    +0.265756,    +0.023942,    +9.787497,     17568920862843,   10.000,    1.423
Gyro   body (rad/s),    -0.005326,    +0.006392,    +0.000000,     17568920722723,    9.965,    1.563
```

Queclink Confidential