**PowerKey Module Description Document**

**1. Overview**

The AP80 series chips have an ONKey (and PowerKey, same below) pin. By configuring and using this pin, you can power on and power off the system. Users can configure this pin to Bypass mode, hard switch (toggle switch) mode, or soft switch (button switch) mode.

From the device's physical design, the ONKey pin is connected to either a toggle-style Power button or a push-button-style Power button. Of course, some devices may not have a Power button, in which case the ONKey can be configured to Bypass mode.

PowerKey startup is automatically controlled by hardware, meaning that once PowerKey is effectively triggered, the system automatically powers on. PowerKey shutdown is controlled by software, meaning that once the system detects a PowerKey shutdown trigger, the software shuts down the system.

**2. PowerKey Functions**

PowerKey is primarily used for system shutdown and power-on. When introducing the functions of the PowerKey, we first introduce a system mode: PowerDown mode. In PowerDown mode, all system components except the LP module are powered off, and the system's standby power consumption is approximately 20uA. When the user wants the system to be in standby mode, they can put the system into PowerDown mode to reduce power consumption. In PowerDown mode, the system can only be awakened (booted) via the RTC and PowerKey.

The functions of the PowerKey are as follows:

  When the system is first powered on, the PowerKey's state determines whether to boot up or shut down：

Some devices have a Power button. When the switch is in the off position, if the device is powered on by inserting a battery at this time, the system should not boot up. After inserting the battery, the system will only boot up when the user flips or presses the Power button.

  When the system is running, the user can use the PowerKey to shut down the system and enter PowerDown mode:

When the system is running, if the user flips or presses the Power key, the system will enter standby mode after detecting the PowerKey shutdown action.

  When the system is in PowerDown mode, the system can be awakened and continue running by using the PowerKey:

When the system is in standby mode, the user can restart the system by flipping or pressing the Power key again.

**3. PowerKey Usage**

**3.1. PowerKey Connection**

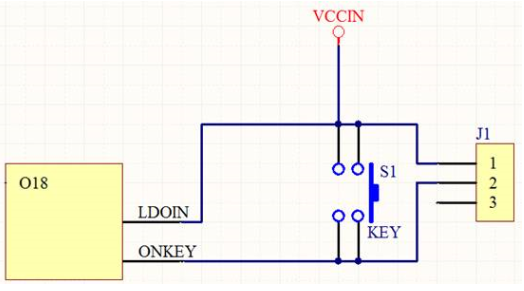


Figure 1 PowerKey Hardware Connection Reference Diagram

The above figure shows the connection diagram of PowerKey in hardware design. S1 represents the pushbutton switch, and J1 represents the toggle switch. S1 and J1 are connected in parallel. In actual products, only one of them is required, i.e., either S1 or J1 can be retained.

PowerKey can be configured in three modes:

  Hard Switch/Toggle Switch Mode

  Soft Switch/Button Switch Mode

  ByPass Mode

**3.2. PowerKey's Hard Switch Mode**

As shown in Figure 1, if only J1 is retained, this connection configuration represents PowerKey's hard switch mode. When the J1 switch is set to pins 1 and 2, it is on; when set to pins 2 and 3, it is off.

**3.3. PowerKey Soft Switch Mode**

As shown in Figure 1, if only S1 is retained, this connection configuration represents the PowerKey soft switch mode. Holding down S1 for several seconds and releasing it indicates the switch is on. Holding down S1 again for several seconds and releasing it indicates the switch is off. This is the difference between soft switch and hard switch modes.

**3.4. PowerKey Soft Switch Mode**

As shown in Figure 1, if S1 is removed and pins 1 and 2 of J1 are directly shorted, this is the PowerKey ByPass mode. ByPass mode means that PowerKey is not used externally. The ONKEY pin is directly connected to a high voltage level.

**3.5. PowerKey Initialization**

The above sections have introduced several modes of PowerKey: hard switch, soft switch, and ByPass mode. In these modes, the way the switch is operated differs. For example, the hard switch mode uses a toggle switch, the soft switch mode involves pressing and releasing, and the Bypass mode has no switch operation at all. Besides these differences, what else are these different modes related to? It's the initialization of PowerKey!

When initializing PowerKey, you must specify which mode to use, and the specified mode must correspond to the actual hardware connection mode externally. For example, if the external connection uses a soft switch configuration, then the PowerKey must be initialized using the soft switch mode:

SysPowerKeyInit(POWERKEY\_MODE\_PUSH\_BUTTON, 2000); //2s

If the external connection uses a ByPass configuration, then the PowerKey must be initialized using the ByPass mode:

SysPowerKeyInit(POWERKEY\_MODE\_BYPASS, 0);

In the SysPowerKeyInit function, the first parameter is the selection of the PowerKey mode, and the second parameter is the time parameter, which is used for debouncing on the PowerKey hardware circuit or to prevent false triggering.

  For the hard switch mode, the time parameter is recommended to be set to 500, representing 500ms

  For soft switch mode, the time parameter is recommended to be set to 2000, representing 2000ms, meaning that the button must be held down for 2 seconds before releasing it to be effective. If the button is released immediately after being pressed for 1 second, it will not trigger the PowerKey.

During system initialization, it is essential to initialize the PowerKey; otherwise, an 8-second system reset issue may occur.

**3.6. PowerKey Detection**

As mentioned in the previous overview, the PowerKey shutdown is initiated by the software after detecting the PowerKey shutdown trigger action. Therefore, PowerKey detection must be implemented in the software.

PowerKey detection is typically performed within a periodic timer interrupt, allowing for regular monitoring of the PowerKey.

Additionally, if PowerKey detection is performed within a software while loop, and the code fails to execute due to software runaway, the system may be unable to shut down. However, as long as the hardware is functioning properly, interrupts will always be generated and executed, thereby preventing issues where PowerKey cannot be detected due to software runaway.

The following code demonstrates PowerKey detection within the timer1 periodic interrupt.

159 \_\_attribute\_\_((section(".driver.isr"))) **void** Timer1Interrupt(**void**)

160 {

161     Timer1IntClr();

162     SystemPowerOffDetect();

163 }

164

165 **void** SystemPowerOffDetect(**void**)

166 {

167 #ifdef USE\_POWERKEY\_SLIDE\_SWITCH

168 #define SLIDE\_SWITCH\_ONTIME 500

169     **static uint16\_t**  slide\_switch\_pd\_cnt = SLIDE\_SWITCH\_ONTIME;// Debounce time 0.5s，See PowerKeyDetect() description

170     **if**(PowerKeyDetect())

171     {

172         **if**(slide\_switch\_pd\_cnt-- == 0)

173         {

174             /\*if slide switch, power down system directly\*/

175             APP\_DBG("PowerKeyDetect->go to PowerDown\n");

176             SysSetWakeUpSrcInPowerDown(WAKEUP\_SRC\_PD\_POWERKEY);

177             SysGotoPowerDown();

178             **while**(1);

179         }

180     }

181     **else**

182     {

183         slide\_switch\_pd\_cnt = SLIDE\_SWITCH\_ONTIME;

184     }

185 #endif

186

187 #ifdef USE\_POWERKEY\_SOFT\_PUSH\_BUTTON

188     **if**(PowerKeyDetect())

189     {

190         **if**(gSys.NextModuleID != MODULE\_ID\_POWEROFF) /\*MSG\_COMMON\_CLOSE only need send once\*/

191         {

192             /\*if use push button, send message, for task's saving info.\*/

193             APP\_DBG("PowerKeyDetect->send message common close\n");

194             gSys.NextModuleID = MODULE\_ID\_POWEROFF;

195             MsgSend(MSG\_COMMON\_CLOSE);

196         }

197     }

198 #endif

199 }