AN2774
Application note

Low power standby management based on MCU

Introduction

The number of electrical products used today in private homes and offices is constantly increasing. Most of these, such as audio/video products and office equipment, consume considerable amounts of electrical power while on standby (when they are not being used but are still connected to the mains supply). This constitutes a waste of energy and money, and has a large impact on the environment. Today, many products required to run in standby mode must adhere to certain regulations that fix stringent limits as to their energy consumption.

This document describes a smart power-management method for electrically-powered devices during their standby operation, making the products compliant with the new norms on power saving. The system architecture is based on a low-power microcontroller which, during standby mode, can significantly reduce energy consumption by disconnecting most parts of the system from the power supply. Only the low-power microcontroller and the system's wake-up sources are permanently powered. When the microcontroller receives an input from one of the possible wake-up sources, all of the circuitry in the system is switched on.

The first part of this document describes a power management system for general-purpose usage, independent of the product. This system can be easily adapted and embedded in consumer or office equipment and home appliances. The second part of the document describes an example of standby management for consumer applications.

Contents

List of figures

1 Normative standby mode consumption

- Standby mode: refers to products when they are switched off but still connected to the mains supply and not performing their normal activity.
- Active mode: refers to products when they are switched on and functioning.

Since many devices stay in standby mode for extended periods of time, an energy cut during this time can greatly reduce energy consumption.

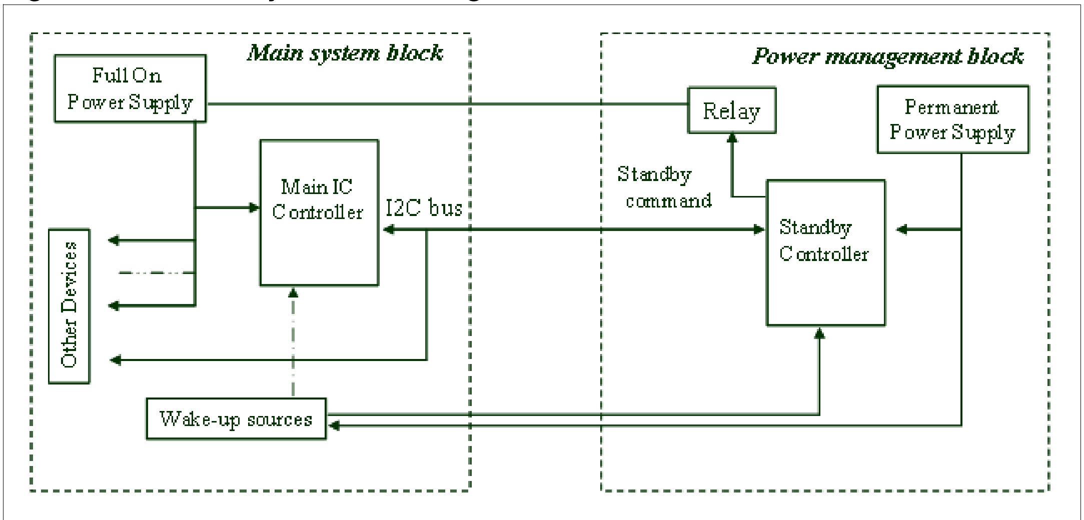
The current specifications from ENERGY STAR® set the amount of power consumed when the device is in standby mode to less than or equal to 1 to 2 W, depending on the type of device. For audio and video products, this value is set to 1 W.

2 System architecture overview

A smart power management control system is necessary to design systems with sustainable principles. This is achieved by adding to the main system a general-purpose low-power microcontroller with specific firmware for power control.

Figure 1 shows the block diagram of a generic system that could be an audio or video device, a piece of office equipment or a home appliance. The system is made up of a main block and a power management block.

Figure 1. Overall system block diagram



The main system block performs the primary activity. Its hardware topology differs from one product to another, but each product has a main IC with an I^2C bus and at least one wake-up source.

The main IC manages each system function, interfacing with the standby controller, wake-up sources and other devices.

The wake-up sources are defined as events that can push the standby controller to exit the standby mode and go back to active mode. Common wake-up sources include the remote controller receiver, the keyboard, VGA and HDMI inputs, and so on. A wake-up source could also be a timer event, used to record or start a special operation or used for periodical updates.

Note: During active mode the wake-up sources interface with the main IC to perform the functions related to the command.

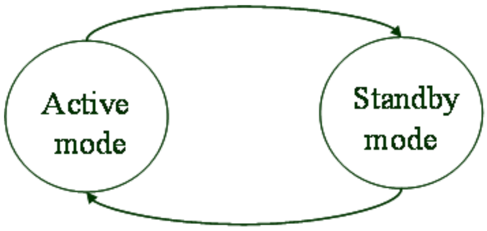
The power management block is a general-purpose block independent of the main system. This block's core is a low-power microcontroller typically used as a standby controller. It receives inputs from the main IC and wake-up sources, switching on and off through the relay the full *on* power supply block that supplies most of the main system parts. During standby mode, the low-power microcontroller and wake-up sources are powered by the auxiliary power supply.

3 System operation

The system functions in active mode (when the main IC and other devices are powered) and in standby mode (when only the low-power microcontroller and the wake-up sources are powered).

The default state after reset is the active mode.

Figure 2. State transition



3.1 Active mode

In active mode, each part of the system gets powered as soon as the system is plugged to the mains supply. The main IC manages each function. It detects user inputs from the wake-up sources such as the remote controller and front panels. It also interfaces with other devices and the low-power microcontroller using the I^2C bus, where the main IC is configured as master and the other devices (low-power microcontroller included) are configured as slave.

In active mode, the main IC updates the low-power microcontroller’s registers to program, for instance, new timer events, or to send the standby command request.

Once the standby controller receives a power-off command from the main IC, it switches to standby mode.

3.2 Standby mode

When the low-power microcontroller receives the standby command from the main IC, the system switches to standby mode, thereby switching off the relay connected to the full *on* power supply. In standby mode, only the low-power microcontroller and wake-up sources are powered on. During this phase, the low-power microcontroller enters a *halt* mode while it waits for external inputs or timer events. Once one of these events has occurred, the low-power microcontroller exits the *halt* mode, decodes the signal received and switches the whole system on through the relay. The type and function of wake-up event is communicated to the main IC through the I^2C bus to perform the related operation. At this stage, the system is in active mode and each function runs normally.

4 Application example

This section describes how to develop a standby management control system for consumer applications, using an 8-bit microcontroller. A REva starter kit from Raisonance (ST7FLITE-SK/RAIS) interfaced with the ST7FOXA0 daughter board (ST7FOXA0-D/RAIS) is used as the development tool.

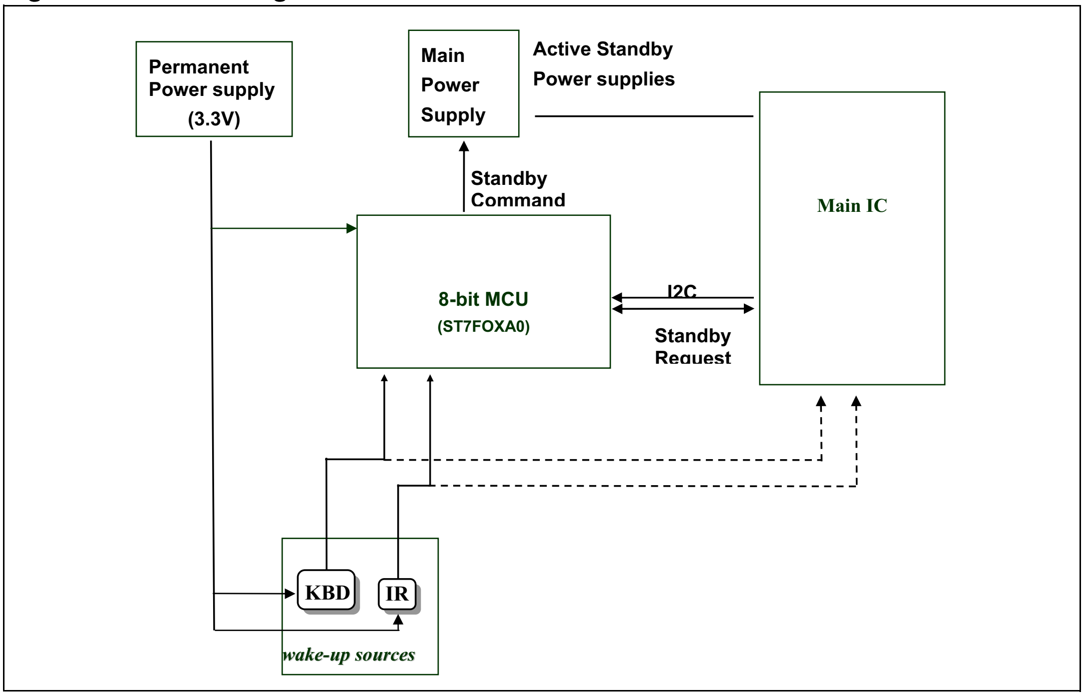
The main components required for development are:

- an IR receiver
- a remote controller based on the RC5 protocol
- a push button
- an I^2C bus
- a LED.

5 Block diagram

Figure 3 shows the block diagram of a generic consumer system used as an example.

Figure 3. Block diagram



The system's core is the ST7FOXA0, an 8-pin and 8-bit microcontroller. It interfaces with the main IC through the I^2C where the 8-bit microcontroller is configured as slave.

The system's wake-up sources are the IR receiver for the remote controller (IR block) and a front panel key (KBD block). In the diagram above, a dotted line is shown going from the IR and KBD blocks to the main IC because during active mode the main IC receives commands from these blocks to perform certain operations.

The 8-bit microcontroller and wake-up sources are powered at 5 V. They are the permanently-powered parts of the system. The main IC is powered by a main power supply switched on and off by the 8-bit MCU according to the standby command.

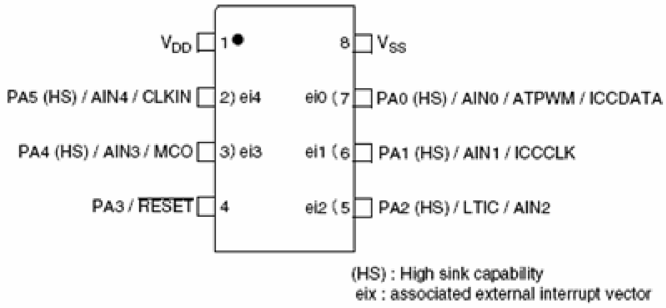
5.1 Features of the ST7FOXA0 microcontroller

The ST7FOXA0 is a member of the ST7 microcontroller family. All ST7 devices are based on a common industry-standard 8-bit core, featuring an enhanced instruction set.

The device is positioned at the entry level of the 8-bit microcontroller range providing an attractive cost while at the same time embedding the most advanced features. The microcontroller is available in SO8 and DIP8 packages.

Figure 4 shows the pinout of the microcontroller.

Figure 4. ST7FOXA0 package pin out



The device embeds the following features.

- 8 pins.
- up to 5 bi-directional I/O lines plus 1 additional output.
- 2 Kbytes single voltage extended Flash memory.
- two timers.
- 10-bit ADC with 5 input channels.

In the application example the pins are configured as shown in *Table 1*.

Table 1. Pinout description

Pin number/name	Pin name	Description
Power		
1 / V_{DD}	V_{DD}	Permanent power supply fixed at 5 V.
8 / V_{SS}	Vss	Ground
4 / PA3	Rst	Reset input
MCU interface		
6 / PA1	I^2C_CLK	Clock line for I^2C communication. I/O pin configured as external input interrupt.
7 / PA0	I^2C_Data	Data line for I^2C communication. I/O pin configured as external input interrupt/output according Tx and Rx operation.
Infra-red receiver		
3 / PA4	IR_IN	Infrared receiver input. I/O pin configured as external input interrupt. Connected to the IR receiver like Vishay TSOP34836.
Front panel command		
5 / PA2	KEY	Push button to switch the system on/off. I/O pin configured as external input interrupt.
Standby control		
2 / PA5	OUT	Output control for the main power supply connected to the LED to display the system status, I/O pin configured as output.

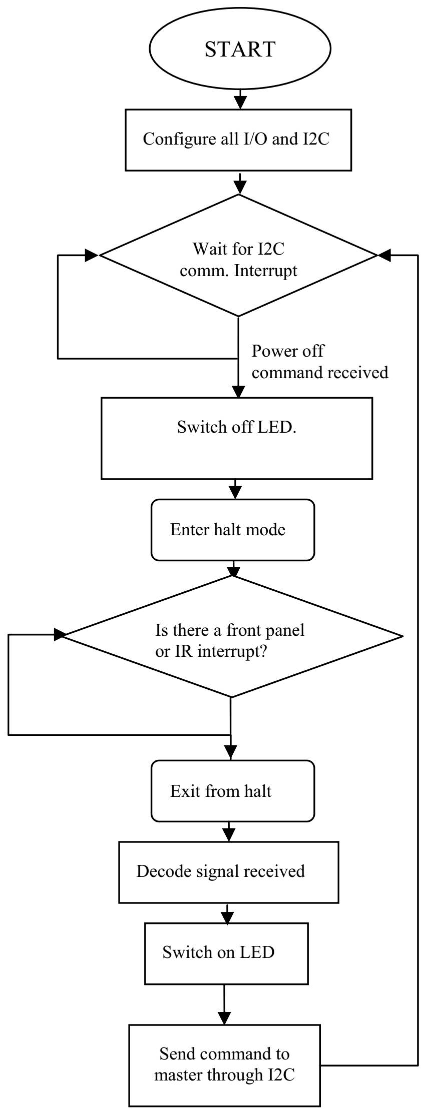
6 Firmware implementation

All source files are written in 'C' language and the application uses the ST7 software library functions. The microcontroller features the following.

- Emulated low-speed I^2C slave interface communication.
- Remote controller receiver and decoder wake-up event.
- Front panel wake-up event.
- Main power supply command with programmable polarity.

Figure 5 shows the application flow.

Figure 5. ST7 program flow



6.1 I^2C interface communication

The 8-bit microcontroller interacts with the main IC through an I^2C bus. The main IC is the master and the 8-bit microcontroller is configured as slave. The ST7FOXA0 does not have a hardware I^2C peripheral and as such is emulated by software. Pins PA0 and PA1 emulate I^2C_Data and I^2C_CLK respectively.

The baud rate of the I^2C is relatively slow, reaching a speed of up to 50 kHz.

The main IC writes the power-off command and reads the wakeup sources and the related command from the 8-bit microcontroller registers through the I^2C.

6.1.1 I^2C slave address

Table 2. Standby controller I^2C address

Write address	Read address
0x50	0x51

6.1.2 I^2C command

The master and slave devices exchange data through the I^2C bus according to the following structure:

- standby command communication from master to slave
  [S] + [0x50|Ack] + [0x01|Ack] + [P]
- exit from standby communication from slave to master
  [S] + [0x51|Ack] + [K|Ack] + [P]

K = 0x00: none

0x01: push button standby

T: remote standby

Note: Value of remote controller command: S=START, P=STOP, Ack=Acknowledge.

6.2 Remote controller receiver

An external interrupt input (IRIN) of the microcontroller is dedicated to decoding of the remote controller code. The decoded protocol implemented is the RC5. *Figure 6* shows a typical pulse train of the RC5 message.

The first two bits (S1 and S2) are the start bits (always "1"). Bit 3 (T) is a toggle bit. If any one command is pressed twice, this bit does not change; on the other hand, if the command is different from the previous one, this bit toggles.

Bits 4 to 8 of the RC5 address should be the same for all selected codes. Bits 9 to 14 of the RC5 command depend on the command selected from the remote controller.

Figure 6. RC5 frame format

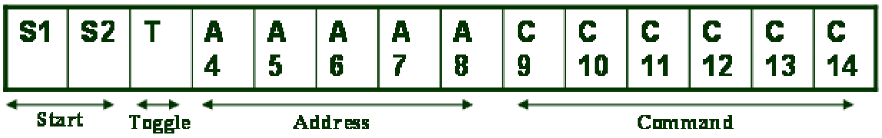


Figure 7. Bi-phase coding

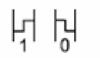
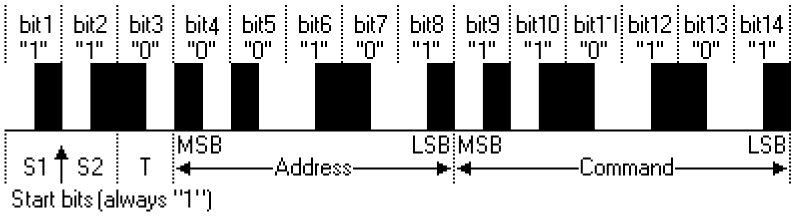


Figure 8. RC5 pulse train



More than one command can be selected to exit standby mode. These commands are reported in *Table 3*.

The code decoding routine is implemented using PA4 configured as an external interrupt and the OVF (overflow) interrupt from the ART (auto reload timer). The port interrupt starts signal acquisition, thereby starting the timer; the OVF timer interrupt samples the bit of the RC5 frame. Each OVF happens at 1/4 of a bit length and it samples its value. If a remote controller source wakes up the system, the code is stored in the dedicated register and sent to the main IC via the I^{2}C.

6.2.1 RC5 remote controller commands

The remote controller RC5 address used in this example is 00000. The commands reported in *Table 3* are used to exit the system from standby.

Table 3. Remote controller commands

Command	Key pressed
0x01	KeyPressed_one
0x02	KeyPressed_two
0x03	KeyPressed_three
0x04	KeyPressed_four
0x05	KeyPressed_five
0x06	KeyPressed_six

Table 3. Remote controller commands (continued)

Command	Key pressed
0x07	KeyPressed_seven
0x08	KeyPressed_eight
0x09	KeyPressed_nine
0x0C	KeyPressed_Power

6.2.2 Remote controller decoding routine

The following is the ART interrupt service routine.

```
/*******************************************************************
Function name : Overflow interrupt
Description   : RC5 decoding management on PA4.
Every OVF happens at 1/4 of bit length and sample its value.
Input param   : None
Output param  : None
*******************************************************************/
@interrupt void ATTimerOVF_Interrupt (void)
{
 extern u8 dummy;
 extern u8 RC5cfg[10];
 extern u8 I2CUserDataByte;
 extern u8 i;

 if (counter<3)
 {
  counter--;
   if (counter==0)
   {
    BitValid--;
    counter=2;
    if (BitValid==1)
      {
         Bitnum--;
         dummy=(PADR&0x10)>>4;
      }
    if (BitValid==0)
    {
      Bitnum--;
         BitValid=2;
         BitLevel=(PADR&0x10)>>4;
         if (dummy!=BitLevel)
         {
            RC5code<<=1;     // if dummy ==1 it have been read "0"
            if (dummy!=0)
            {

             RC5code++;    // it have been read "1"
            }
         }
         else
         {
          Bitnum=2;
          RC5code=0;
          SetBit(PAOR,4);
          counter=1;
         }
      }
```

```
         }
         if (Bitnum==0)
         {
               i=10;
               RC5code = RC5code | 0xFA00;

            while(i)
            {
               if(RC5cfg[i]== (RC5code & 0xff))
               {
                  I2CUserDataByte=RC5cfg[i];
                  DVD_State=STBY_ON;
                  break;
               }
               i--;
               }
               if (DVD_State!=STBY_ON)
               DVD_State=STBY;
               RC5code=0;


            while ((PADR&0x10)>>4 ==0x00);  //Wait for IDLE


            ClrBit (write_ATCSR,1);
            SetBit(PAOR,4);
            counter = 5; // disable the reading of the IR

         }
       }

 ATCSR;
}// end routine
```

6.3 Front panel key

The system is switched ON by way of a push button switch located on the front panel. An external interrupt input is connected to the PA2 pin, and the interrupt sensitivity is set on the rising edge. Once the button is pressed and the system is in standby, the micro exits from halt mode and powers on the whole system.

Note: If the system is in active mode, the microcontroller switches off the system. The push button is detected by the main IC, which sends the standby command to the microcontroller.

6.4 Main power supply command

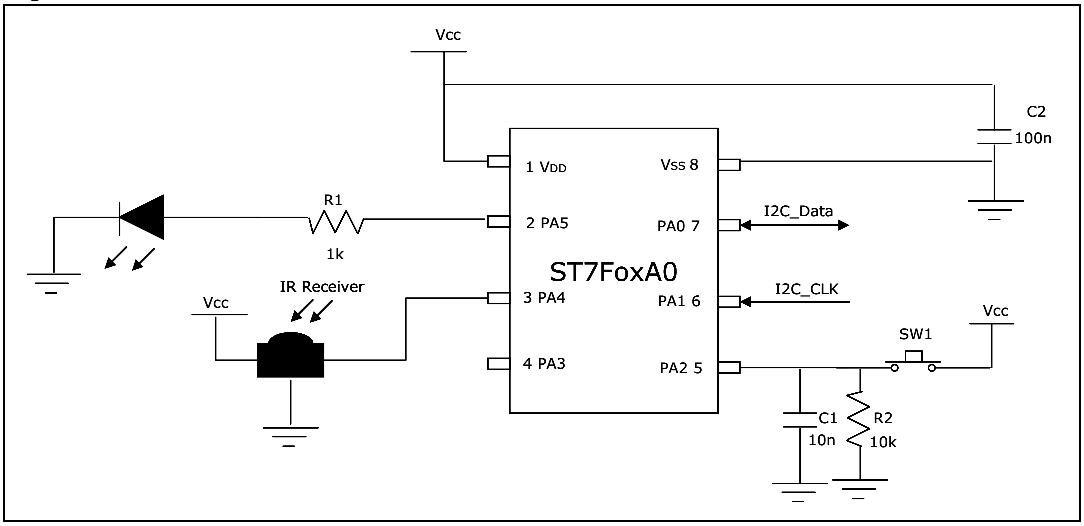
Typically, the microcontroller switches on and off the main power supply driving a relay. In this application the relay is replaced by a LED to display the status of the system.

The LED is directly connected to the PA5 pins of the microcontroller and is configured as an output.

7 Hardware configuration

This application has been developed and tested by interfacing the ST7FOXA0 daughter board (ST7FOXA0-D/RAIS) with the main REva starter kits from Raisonance (ST7FLITE-SK/RAIS). *Figure 9* shows the hardware setup that you can also use to develop this application if you are not using the REva board.

Figure 9. Hardware connection



An external infra-red receiver, such as the Vishay TSOP34836, is connected to the PA4 pin.

The remote controller is a universal one with RC-5 protocol. This board uses the remote controller W/PHI-16 with order code 11/20750 from WORLDCELL-e.

The push button switch SW1 is used as a front-panel command to switch on the system, and is connected to the PA2 pin configured as an external interrupt.

The I^2C peripheral is emulated by software. Pins PA0 and PA1 are used as I^2C_Data and I^2C_CLK respectively. The main IC on the REva board is not connected, and must be interfaced with a generic Master board connected to the I^2C bus.

The LED is directly connected to the PA5 pins of the microcontroller to display the status of the system. It is configured as output and switched on and off by the microcontroller according to the system status:

- LED on when system is in run mode.
- LED off when system is in standby mode.

Table 4. Source code features

Parameter	Value
Code size	900 Byte (1)
Peripherals usage	Auto reload timer
Interrupt usage	ei0, ei1, ei2, ei3 external port interrupt

1. The code has been written in C using the Cosmic Software C compiler version 4.5.5 and the STVD version 4.0.1.

8 References

1. REva board hardware description and user guide: Raisonance Universal Evaluation Board with RLink for Microcontrollers.
2. ST7FOXA0 datasheet.

9 Definitions

Table 5. Acronyms and terms used in this document

IC	Integrated circuit
MCU	Microcontroller unit
I^2C	Inter-integrated circuit
IR	Infrared
RC5	Remote controller protocol from Philips
VGA	Video graphics array. Standard interface for video products
HDMI	High-definition multimedia interface
KBD	Keyboard

10 Revision history

Table 6. Document revision history

Date	Revision	Changes
24-Oct-2008	1	Initial release.