APPLICATION NOTE

AN10221
In-system programming (ISP) with the Philips P89LPC932 microcontroller

Richard Soennichsen

2003 Sep 08

Philips
Semiconductors

PHILIPS

OVERVIEW

The Philips P89LPC932 Microcontroller is programmable via the following methods:

- In-System Programming (ISP)
- In Application Programming (IAP)
- Parallel Programming

Although this document only describes ISP in detail, the differences between each of the programming methods are explained below.

In System Programming occurs when an outside device causes the processor, rather than executing its normal application code, to execute code at a location that contains memory erase and programming routines. This programming takes place with the microcontroller in its normal hardware environment. i.e., soldered on a printed circuit board.

In Application Programming is similar to in system programming described above, but rather than having an external device initiate the programming process the normal application code branches to the memory erase and program routines. Similar to In System Programming this operation takes place with the microcontroller in its normal hardware environment.

Parallel Programming requires an external programming device. In general it can only be accomplished on a part that is not in its normal hardware environment.

In-System Programming (ISP)

As mentioned above the In-System Programming mode allows the microcontroller to be programmed even after it has been soldered in a printed circuit board. This feature can be used to perform firmware updates at the end of the production line. Typical applications include adding calibration information or installation of the latest software release. In-System programming may also be used during product development as a quick and easy way to modify program code.

All P89LPC932 devices are shipped with a factory “boot-loader” which is programmed into the upper 512 bytes of sector seven of the code space. This firmware provides the interface between low-level routines in the device, which perform the requested function (program, erase, etc.), and the serial port. *If the user wishes to utilize ISP, care must be taken to not erase or overwrite the code sector containing the ISP firmware (1C00H to 1FFFH).*

Using In-System Programming Hardware Interface

Figure 1 shows that the only hardware needed to connect the LPC932 to a PC’s RS232 port is a simple RS-232 level shifter, such as the Maxim MAX3322.

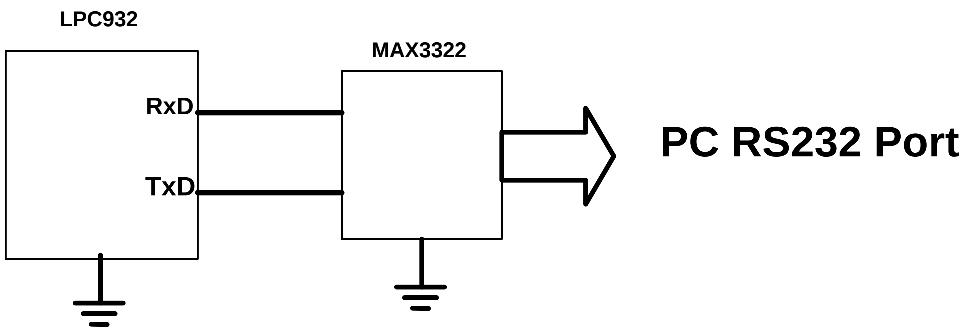


Figure 1

While ISP may be accomplished on a PC running a simple terminal emulation program, it is far easier to use one of the available programs, such as Flashmagic, which incorporate all of the ISP functions.

Entering ISP Mode

There are three ways to enter the ISP mode:

- Via the status bit and boot vector. (Default condition on initial power-up.)
- Through a break detect reset.
- By pulsing the reset pin upon power-up. (Hardware Activation)

Figure 2 shows a flowchart of these alternatives. A more detailed explanation of each follows.

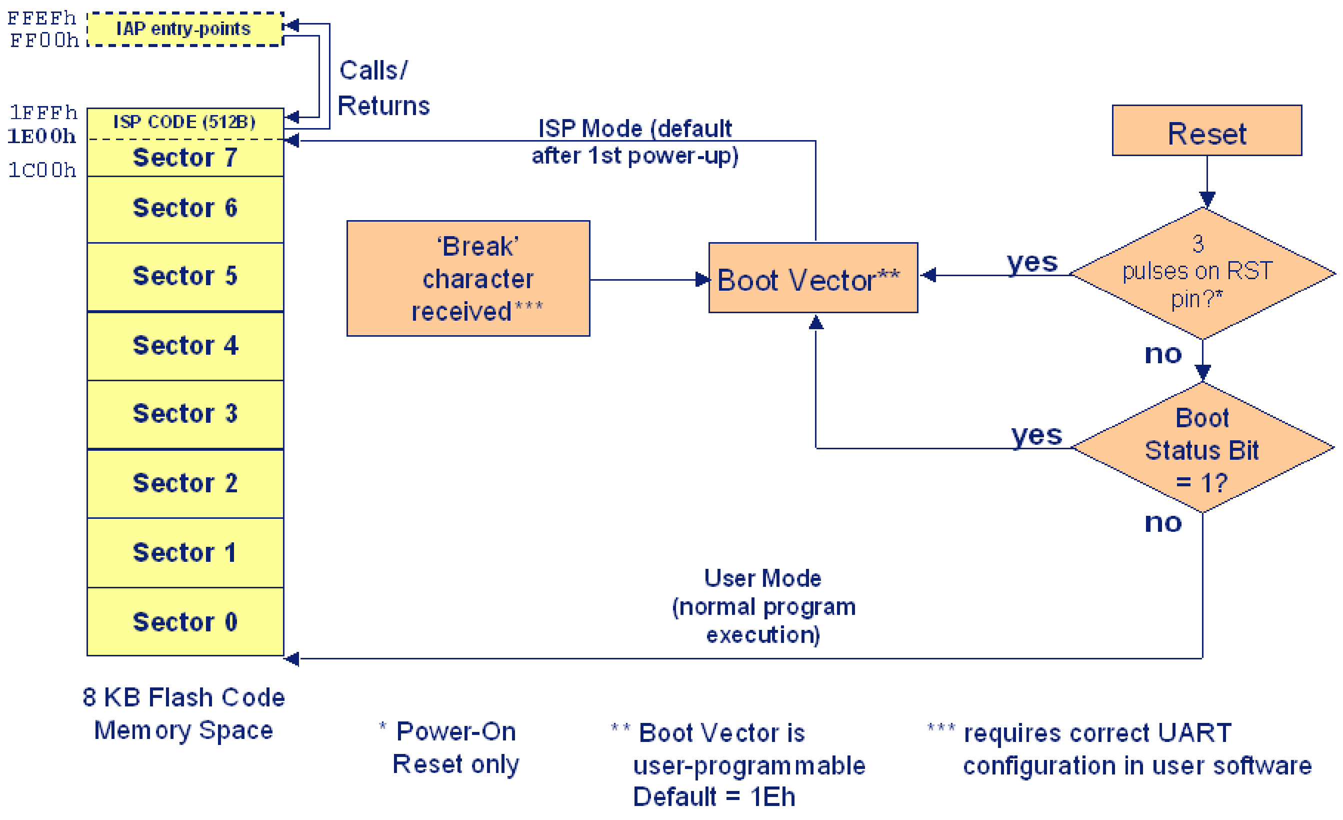

FFEFh
FF00h
IAP entry-points
Calls/ Returns
1FFFh
1E00h
1C00h
ISP CODE (512B)
Sector 7
Sector 6
Sector 5
Sector 4
Sector 3
Sector 2
Sector 1
Sector 0
ISP Mode (default after 1st power-up)
Reset
'Break' character received***
Boot Vector**
yes
3 pulses on RST pin?*
no
Boot Status Bit = 1?
yes
no
User Mode (normal program execution)
8 KB Flash Code Memory Space
* Power-On Reset only
** Boot Vector is user-programmable Default = 1Eh
*** requires correct UART configuration in user software

Figure 2: ISP Entry Diagram

The Status Bit and Boot Vector

Upon reset, entry into ISP mode is controlled by the state of a status bit that is stored in a reserved location in flash memory. Following a reset, the device will examine the contents of the status bit, if it is zero, program execution begins at address 0000H which typically contains user code. If the status bit is not zero, program execution then begins at the address made up of the boot vector, also stored in a reserved flash memory location, concatenated with 00H. When new, parts will have the status bit set and the boot vector programmed to 1EH, Thus upon reset new parts will start execution at address 1E00H; the location of the factory supplied ISP boot loader. For those applications where the user wishes to start execution at 0000H the status bit can be cleared on a parallel programmer when the user code is being loaded. The ISP program itself can reset the status bit. If this is done after application code has been loaded the next reset will result in execution of the user code. The ISP program can also alter the value of the boot vector e.g., to a location where a custom boot loader resides. *However if altered from its initial, factory set, value, 1EH, it will not be possible to execute the factory boot loader. If the boot vector is changed to point to a location which does not contain a boot loader a parallel programmer will be needed to reset it to point to the entry point of an ISP program.*

Break Detect Reset

A second mode of entry is through a UART break detect reset. A break condition is defined as a low on RxD for the length of one frame time. Frame length depends on the particular mode of the UART. In mode one for example a break condition is defined as ten bit times. When a break is issued RxD is typically held low for multiple frame times. The break is reported after the first frame that meets the condition of RxD being held low. *Note: If a break condition is created by pulling RxD low with a mechanical switch, and the part is configured with serial interrupts* *enabled, then the RI (receive flag) must be handled in an interrupt service routine or the part will enter an undefined state.*

The method requires that the *user's code* initialize the UART. This means enabling the UART and enabling break detect reset by setting the break enable bit (EBRR) in SFR AUXR1. Once this is done a break condition on the RxD pin will cause the part to reset and program execution to re-start at the location pointed to by the boot vector. An example of this user code follows.

```
void initialize_ISP (void)
        {
          SCON = 0x50;          //select the Baud Rate Generator as UART baud rate source
          BRGR1 = 0x04;         //9600 BAUD at 11.0592 MHz
          BRGR0 = 0x70;

          BRGCON = 0x03;        //enable BRG

          AUXR1 |= 0x40;        //enable reset on break detect by setting EBRR
        }

void UART_ISR(void) interrupt 11
        {
          RI = 0;               //This is necessary if the break condition is
                                //created by a mechanical switch and serial interrupts
                                //are enabled.
        }
```

Hardware activation

This mode of ISP entry is always available regardless of user code or the state of the status bit. (Assuming that the boot loader code is intact and the boot vector is 1EH) By presenting a timed waveform of low-going pulses (see Figure 3) to the reset pin after power up, the part will begin code execution at the address pointed to by the boot vector. This entry mode has the same effect as having a non-zero status byte.

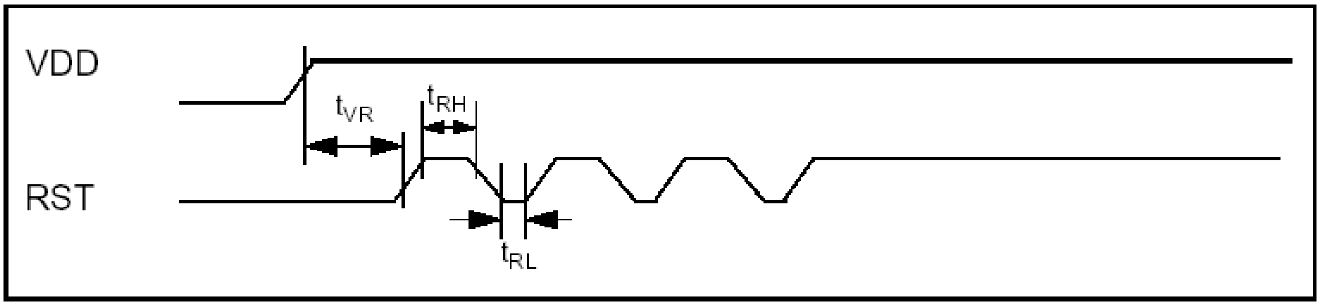


Figure 3

SYMBOL	PARAMETER	MIN	MAX	UNIT
t_{VR}	RST delay from V_{DD} active	50	-	µs
t_{RH}	RST HIGH time	1	32	µs
t_{RL}	RST LOW time	1	-	µs

Table 1: ISP Entry, AC Characteristics

Note: Providing more or less than the required three pulses will result in the device not going into ISP mode.

Hardware Activation of ISP Mode, an Example

By introducing some low-going pulses to the reset pin after power-up we can put the device into ISP mode. This assumes that the boot loader is intact and the boot vector is still set to 1EH.

Figure 1 and Table 1 describe the required waveform. A typical application might have a header on the PCB to accept a pre-made “dongle” that will control power and reset to the P89LPC932. In this example a P87LPC760 microcontroller controls the voltage to the P89LPC932 via a Philips SA57000 voltage regulator and also provides the pulses to the P89LPC932 reset pin.

The code is written so that, after the “dongle” is plugged onto the application, the LED (two-color) will glow red. After the button is pushed, power is applied to the P89LPC932 while it is also being held in reset. After a delay the reset is released, the pulses are sent, and the LED glows green. The P89LPC932 is now in ISP mode. Pressing the button again puts the P89LPC932 back into reset. Pressing the button again repeats the entry process. (The application code is listed at the end of this document.)

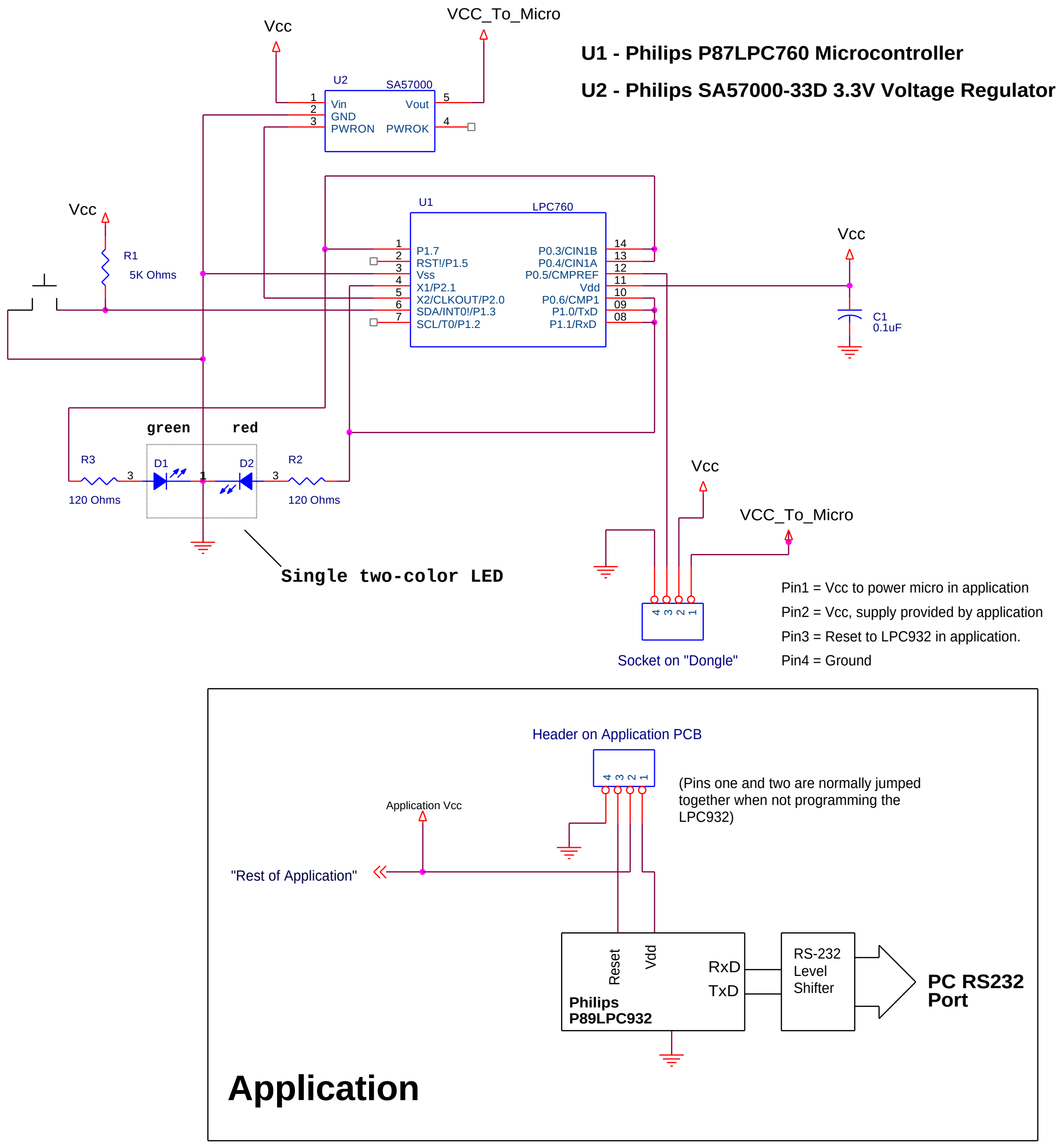


Figure 3: Hardware activation "dongle" Schematic

Alternatively, with a small amount of additional circuitry, a four-pin header with no jumper is all that is necessary in the application.

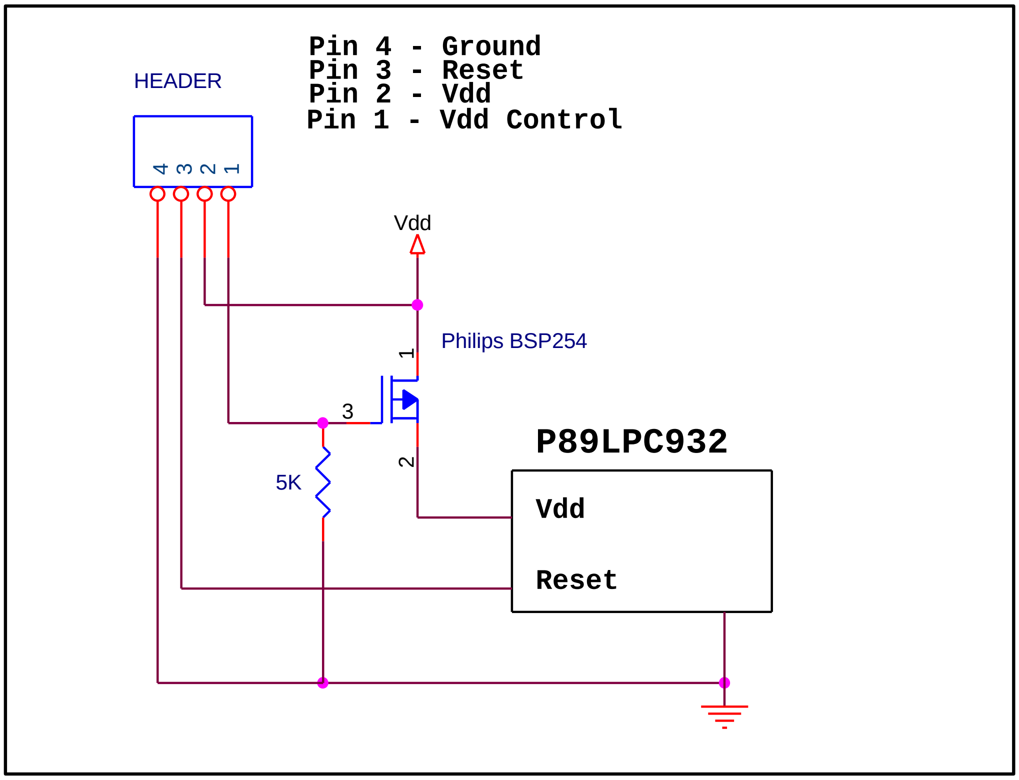


Figure 4: Alternative application Design with a four pin header and no jumper necessary.

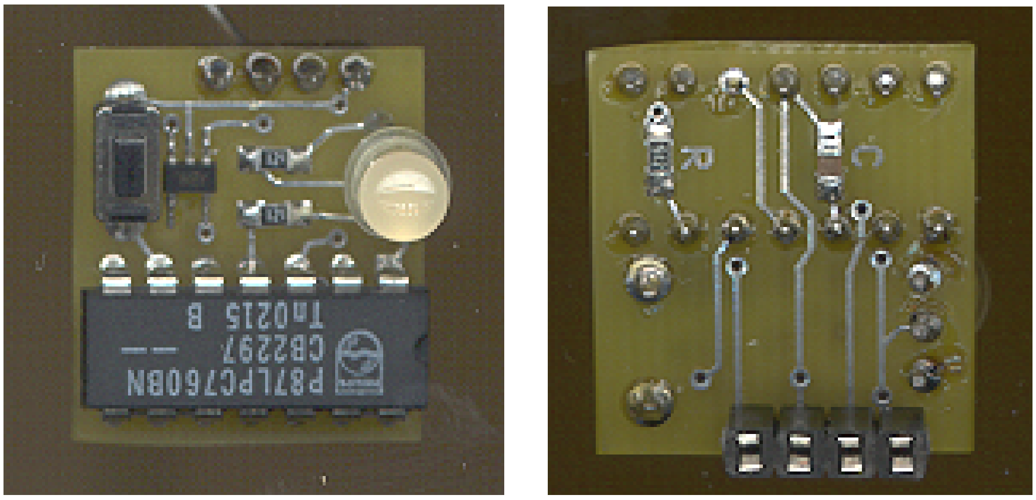
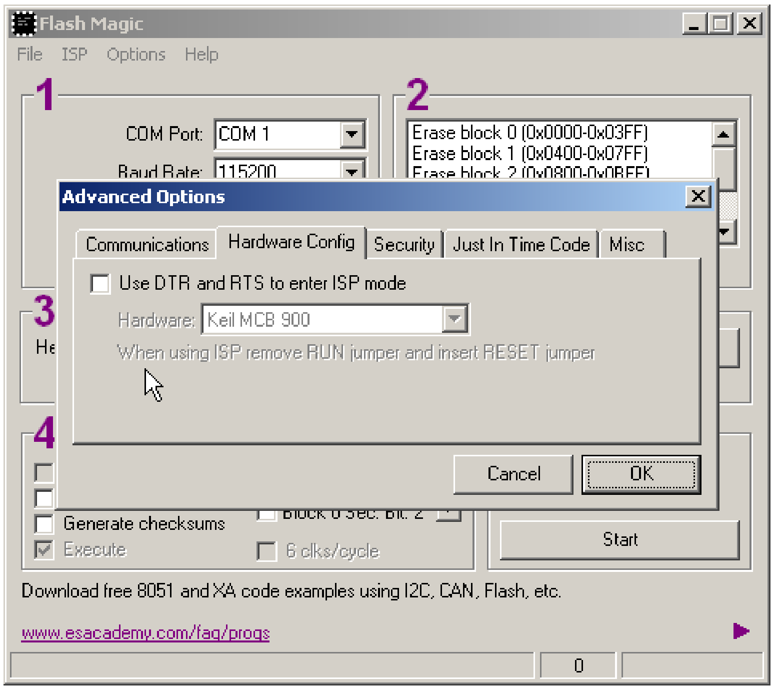


Figure 5: The Hardware Activation "Dongle"

Using Flashmagic with the P89LPC932

Flashmagic is a very popular (and free) utility to program many Philips devices, including the P89LPC932. It is available at http://www.esacademy.com/software/flashmagic/. Upon starting, Flashmagic will attempt to connect to the device selected. It is normal that a "connection error" message appears as there is probably no device present in ISP mode or the settings need to be adjusted. Select the correct COM port for your PC, select the P89LPC932 as the device. Now the device needs to be put into ISP mode, if it is a new device it will already be in ISP mode. If it is not a new device (The status bit is not set) then enter via hardware activation (three pulse) or use the break detect reset.

Before we use the break condition to enter ISP mode Flashmagic must be configured to use this method to initiate ISP. Under the menu options/advanced options/ hardware config the "Use DTR and RTS to enter ISP mode" must be unchecked. (Leaving the box checked configures Flashmagic to use DTR and RTS to produce the signals necessary to enter ISP mode via the hardware activation method. The hardware necessary to implement this must be used, i.e. the Keil MCB 900 evaluation board.)



COM Port: COM 1
Baud Rate: 115200

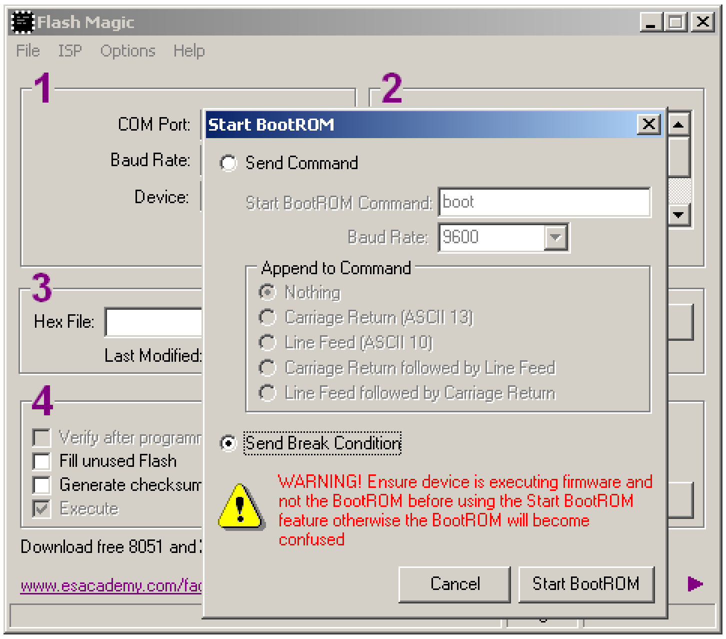
Erase block 0 (0x0000-0x03FF)
Erase block 1 (0x0400-0x07FF)
Erase block 2 (0x0800-0x0BFF)

☐ Use DTR and RTS to enter ISP mode
Hardware: Keil MCB 900
When using ISP remove RUN jumper and insert RESET jumper

☐ Generate checksums
☑ Execute
☐ 6 clks/cycle

Download free 8051 and XA code examples using I2C, CAN, Flash, etc.

Flashmagic will create the break condition for you, select "start bootrom" under the ISP menu and select "send break condition.



Start BootROM

( ) Send Command

Start BootROM Command: boot

Baud Rate: 9600

Append to Command
(●) Nothing
( ) Carriage Return (ASCII 13)
( ) Line Feed (ASCII 10)
( ) Carriage Return followed by Line Feed
( ) Line Feed followed by Carriage Return

(●) Send Break Condition

WARNING! Ensure device is executing firmware and not the BootROM before using the Start BootROM feature otherwise the BootROM will become confused

At this point the device should be in ISP mode. All ISP functions are now available to you including erase, program, read status bit and boot vector, security bits, etc.

Note: By default Flashmagic will protect the bootrom code in sector seven. The user may over-ride this setting in the options/advanced options/ security menu.

Hardware Activation Code ("dongle")

```
/*****************************************************************************************************************
P89LPC932 Hardware Activation code for P87LPC760  based "dongle"
The delays are written for a P87LPC760 running from the internal RC oscillator (6 MHz)

Richard Soennichsen
Philips Semiconductors
September 8, 2003

*****************************************************************************************************************/
#include <Reg762.h>                             //Include file for Raisonance Compiler

#define on 1                                    //Use these if using the jumper header in the application
#define off 0

//#define on 0                  //Use these if powering the P89LPC932 with the MOSFET (alternative design)
//#define off 1

void main(void);                                //Function Declarations
void init(void);
void msec(int x);
void LED(char led);
void ext0(void) interrupt 0;
void pulse(void);
void rled (bit red);
void gled (bit green);

at 0x93 sbit button;                    //Button located on P1.3
at 0x85 sbit reset;                     //Reset control is on P0.5
at 0xA0 sbit vdd;                               //Vdd control is on p2.0
at 0x86 sbit rled0;                     //port pins associated with the RED LED
at 0x90 sbit rled1;
at 0x91 sbit rled2;
at 0xA1 sbit rled3;

at 0x97 sbit gled0;                     //port pins associated with the GREEN LED
at 0x83 sbit gled1;
at 0x84 sbit gled2;

bit led_status=1, pulse_out = 0;                //Initialize LED and pulse control flags

//********  Initialization Routine  *****************************
void init(void)
        {

                P0M1 = 0x00;            //Initialize all ports to push-pull mode
                P0M2 = 0xFF;

                P1M1 = 0x00;
                P1M2 = 0xFF;

                P2M1 = 0x00;
                P2M2 = 0xFF;

                P1M1 |= 0x08;           //except external interrupt zero which is set to be high
                P1M2 &= 0xF7;   //impedance

                LED(1);                 //Turn LED on red
                vdd=off;                //Turn off Vdd to P89LPC932
          reset = 0;            //Hold the P89LPC932 in reset
                EX0 = 1;                //Enable external interrupts
                EA = 1;                 //Global interrupt enable
        }

//************  Main Program Loop  ********************************
void main(void)
{
init();                                 //Initialize

while(1)
        {
                if(pulse_out)           //If the pulse_out flag is set, send pulses
                pulse();                //send pulses
        }
}

//************  LED Control Routine  *********************************
void LED(char led)
        {
                if(led==1)                                      //make red
                        {
                                gled(0);
                                rled(1);
                        }
                else if (led ==0)                       //make green
                        {
                                rled(0);
                                gled(1);
                        }
        }

//************  LED Blinking Routine  ************************
void blink(void)
        {
                char j=0;
                while(j<6)
                {
                    LED(0);
                    msec(200);
                    LED(1);
                    msec(200);
                    j++;
                    }
          }

//*************  External interrupt (button) Service Routine  ***************
void ext0(void) interrupt 0
          {

                    msec(50);                     //De-bounce
                    while(button==0)                        //Wait for button to be released
                              {
                                        msec(10);
                              }

                    if (led_status == 0)
                              {
                                        led_status = 1;     //Turn LED red
                                        pulse_out = 0;      //Don't send pulses
                              }
                    else
                              {
                                        led_status = 0;     //turn LED green
                                        pulse_out = 1;      //send pulses
                              }
                    LED(led_status);                        //Change LED accordingly

          }

//*************Pulse Generation Routine  *****************************
void pulse(void)

          {
                              char j;

                              reset = 0;          //Hold the P89LPC932 in reset
                              vdd = on;           //Turn on Vdd to the P89LPC932
                              blink();                      //Blink the LED

                              msec(10);           //wait for Vdd to stabilize

                              reset = 1;          //release reset
                              for (j=0; j<10; j++);  //Pulse high time
                              reset = 0;
                              for (j=0; j<5; j++);          //Pulse low time

                              reset = 1;          //Repeat two more times
                              for (j=0; j<10; j++);
                              reset = 0;
                              for (j=0; j<5; j++);

                              reset = 1;
                              for (j=0; j<10; j++);
                              reset = 0;
                              for (j=0; j<5; j++);
                              reset = 1;
                      msec(2);

                              LED(0);                       //Turn LED green
                      while (pulse_out);          //Wait here until the button is pressed again
                      vdd = off;                            //Button pressed, turn off Vdd to P89LPC932
                      reset = 0;                    //Hold the P89LPC932 in reset
                    }
//******  Generic Delay  *****************************
```

```
void msec(int x)                        //P87LPC760 running from internal RC oscillator (6MHz)
        {
                int j=0;
                while(x>=0)
                        {
                                for (j=0; j<300; j++);
                                x--;
                        }
        }
void gled (bit green)
        {
                if(green)
                        {
                                gled0 = 1;
                                gled1 = 1;
                                gled2 = 1;
                        }
                else
                        {
                                gled0 = 0;
                                gled1 = 0;
                                gled2 = 0;
                        }
        }

void rled (bit red)
        {
                if(red)
                        {
                                rled0 = 1;
                                rled1 = 1;
                                rled2 = 1;
                                rled3 = 1;
                        }
                else
                        {
                                rled0 = 0;
                                rled1 = 0;
                                rled2 = 0;
                                rled3 = 0;
                        }
        }
```

In-System Programming with the P89LPC932 AN10221

Revision history: **2003-09-08** **Revision 02**

Contact information

For additional information please visit
http://www.semiconductors.philips.com. Fax: **+31 40 27 24825**

For sales offices addresses send e-mail to:
sales.addresses@www.semiconductors.philips.com

Date of release: 09-03
Document order number: 9397 750 12027

Let's make things better.

Philips
Semiconductors

PHILIPS