Semiconductors

AN10356

Entering ISP mode from user code

Rev. 01 — 04 February 2005 Application note

Document information

Info	Content
Keywords	ARM ISP, bootloader
Abstract	Entering ISP mode is normally done by sampling a pin during reset. This application note describes a method whereby ISP mode may be entered while running in user code.

PHILIPS

Revision history

Rev	Date	Description
01	20050204	Initial version

Contact information

For additional information, please visit: http://www.semiconductors.philips.com

For sales office addresses, please send an email to: sales.addresses@www.semiconductors.philips.com

1. Introduction

In-System programming (ISP) is a method of programming and erasing the on-chip flash or RAM memory using the bootloader software and a serial port. The part may reside in the end-user system. The flash bootloader provides an In-System Programming interface for programming the on-chip flash or RAM memory. This bootloader is located in the upper 8 kB of flash memory, it can be read but not written to or erased.

In many cases it may be desirable to enter ISP mode without resetting the device, while running user code. A method of doing this, which involves only a small amount of code, is described in this document.

2. LPC2000 ISP overview

The flash bootloader code is executed every time the part is powered on or reset. The loader can execute the ISP command handler or pass execution to the user application code. A LOW level, after reset, at the P0.14 pin is considered the external hardware request to start the ISP command handler. The bootloader samples this pin during reset.

Assuming that a proper signal is present on the X1 pin when the rising edge on the Reset pin is generated, it may take up to 3 ms before P0.14 is sampled and the decision on whether to continue with user code or ISP handler is made. If P0.14 is sampled LOW and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored. If there is no request for the ISP command handler execution (P0.14 is sampled HIGH after reset), a search is made for a valid user program. If a valid user program is found then the execution control is transferred to it. If a valid user program is not found, the auto-baud routine is invoked. As Pin P0.14 is used as the hardware request for ISP, it requires special attention. Since P0.14 is in high impedance mode after reset, it is important that the user provides external hardware (a pull-up resistor or other device) to put the pin in a defined state. Otherwise unintended entry into ISP mode may occur.

Fig 1 shows the boot sequence of the LPC2100 devices.

Fig 2 shows the boot sequence of LPC2000 devices (Bootloader revisions 1.61 and later, not applicable to the LPC210x devices).

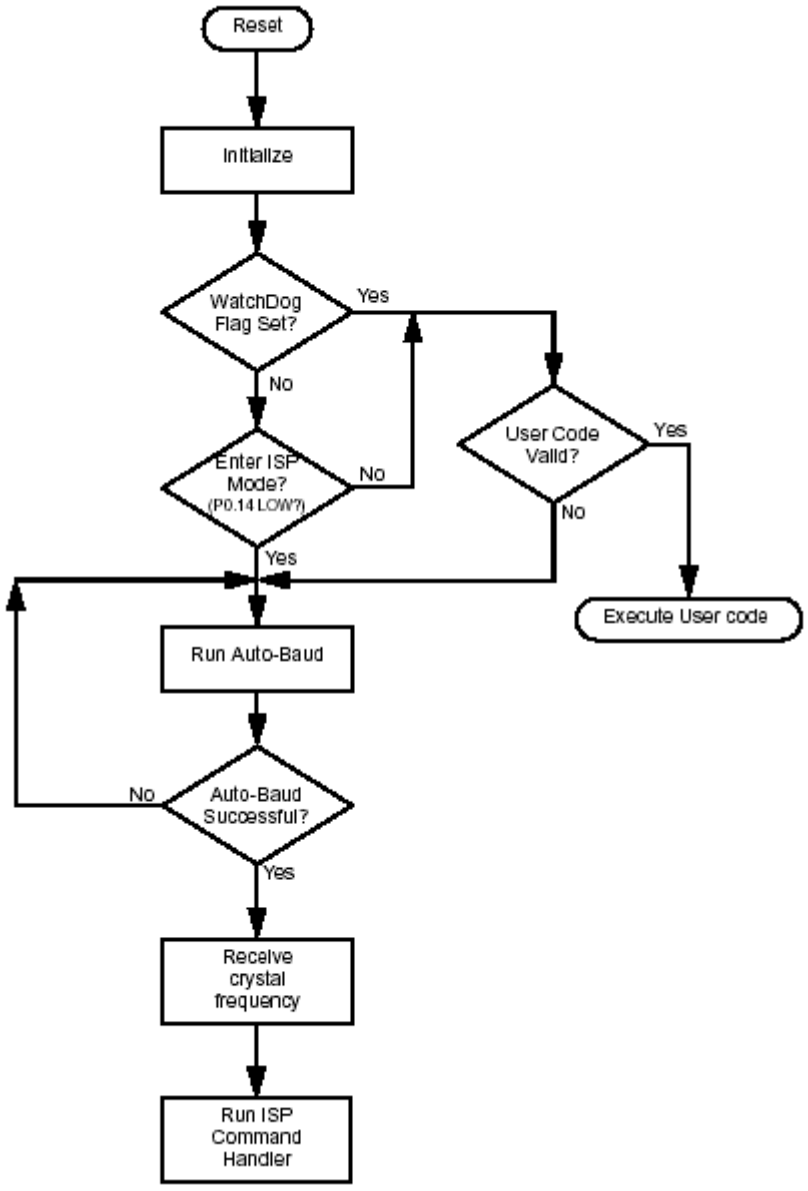

Fig 1. Boot process flowchart (LPC210x devices)

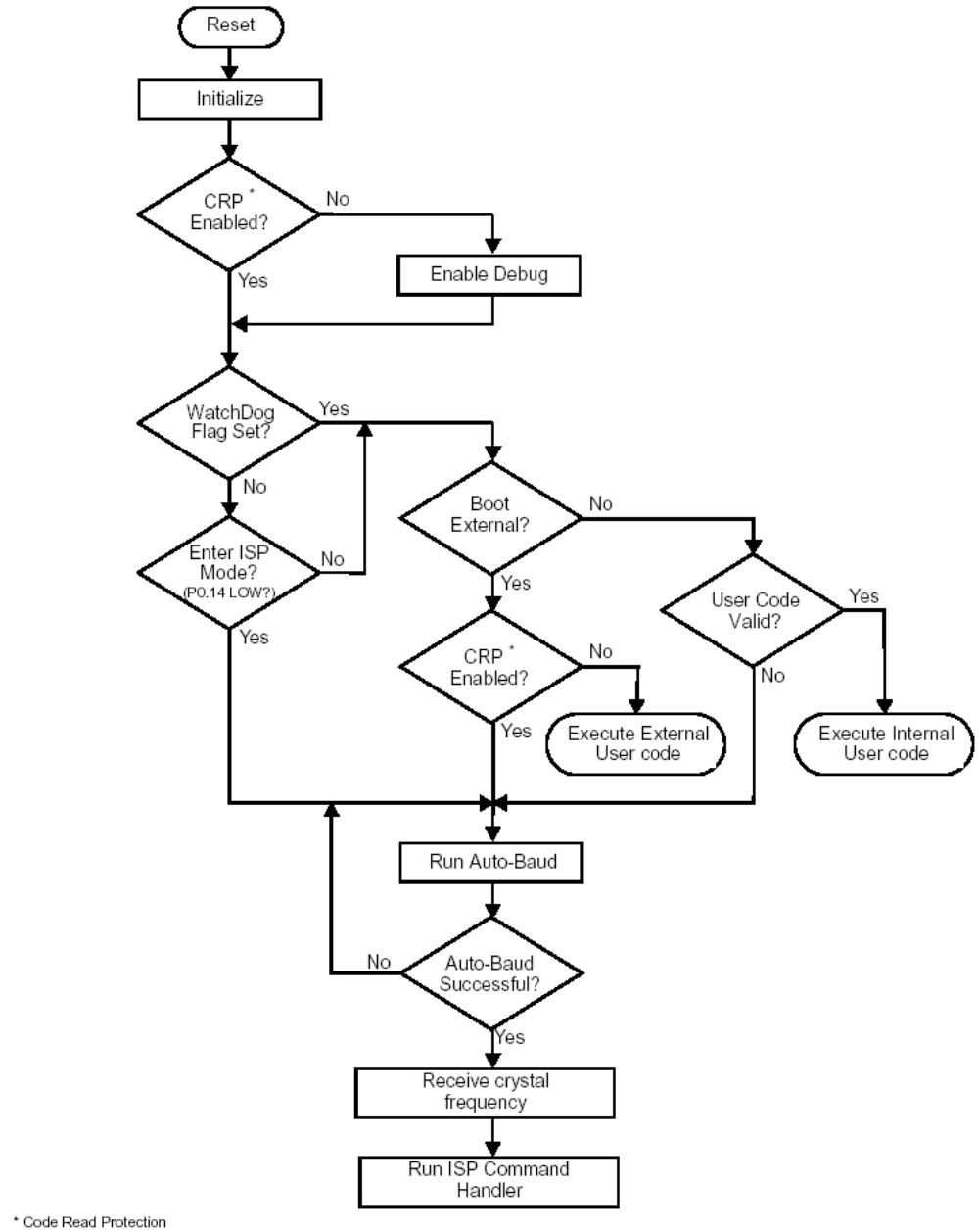

Fig 2. Boot Process flowchart (LPC2000 Bootloader revisions 1.61 and later, not applicable to the LPC210x devices)

3. Entering ISP Mode from User Code

The steps involved in entering ISP mode from user code are as follows:

- Connect/Configure the RXD0 and TXD0 (UART 0 transmit and receive pins). This is done by entering the appropriate values in the PINSEL0 register.
- Configure P0.14 as an output pin.
- Clear P0.14 (Set low).
- If the PLL is connected, disconnect it (Recommended).
- Set the Peripheral Bus Divider to ¼ if needed.
- Re-Map the interrupt vectors to the boot block.
- Invoke the bootloader by calling a function that is located at the bootloader entry point, i.e. the reset vector at 0x00.

The code to perform the operations listed above is as follows:

```
#define MEMMAP   (*((volatile unsigned int *) 0xE01FC040))
#define IODIR0   (*((volatile unsigned int *) 0xE0028008))
#define IOCLR0   (*((volatile unsigned int *) 0xE002800C))
#define PINSEL0  (*((volatile unsigned int *) 0xE002C000))
#define VPBDIV   (*((volatile unsigned int *) 0xE01FC100))
#define PLLCON   (*((volatile unsigned int *) 0xE01FC080))
#define PLLFEED  (*((volatile unsigned int *) 0xE01FC08C))

void (*bootloader_entry)(void);
unsigned long temp;

void init(void)
  {
temp = PINSEL0;
/* Connect RXD0 & TXD0 pins to GPIO */
PINSEL0 = temp & 0xFFFFFFF3;

/* Select P0.14 as an output and P0.1 as an input */
temp = IODIR0;
temp = temp | 0x4000;
temp = temp & 0xFFFFFFFD;
IODIR0 = temp;

/* Clear P0.14 */
IOCLR0 = 0x4000;

/*
   Disconnect PLL if you want to do ISP at crystal frequency.
   Otherwise you need to pass the PLL freq when bootloader goes in
   ISP mode.
   cclk = crystal when PLL is disconnected
   cclk = PLL freq when PLL is connected.

   Disconnecting the PLL is recommended.   */

   *pllcon = 0x0;
   *pllfeed = 0xAA;
   *pllfeed = 0x55;
```

```
/*
   Set the VPB divider to 1/4 if your application changes the VPBDIV value.
   The bootloader is hard-coded to use the reset value of VPBDIV register
   *vpbdiv = 0x0;
*/

/* Map bootloader vectors */
   MEMMAP = 0x0;

/* Point to bootloader entry point i.e. reset vector 0x0 */

   bootloader_entry = (void (*)(void))(0x0);
}
/*
   Invoke the bootloader
   The bootloader will read pin P0.14 to detect if ISP is forced
   Since P0.14 is configured as an output and set to 0, the bootloader
   will go in ISP mode.
*/
int main(void)
  {
    init();
   while(1)
   bootloader_entry();
  }
```

4. Disclaimers

Life support — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no licence or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

5. Contents

Date of release:04 February 2005

Document number: 9397 750 14665

Published in The Netherlands