2.1.2 Activity indicator LED

While the microcontroller has entered into the Ethernet bootloader, it will activate the Activity Indicator LED. This LED shows that the bootloader is currently running.

2.1.3 Software handshake

In some applications, the system is required to share existing pins or limited pins, the Ethernet bootloader will have to perform some form of handshaking to ensure necessary conditions are met before entering.

2.1.4 Channel selection

In many applications, serial communication is preferred because it saves on I/O pins.

In this application note, the focus will be on the Ethernet communications channel.

All bootloaders have some form of Host-Slave relationship. Usually, the host device (in this case the windows workstation), will be initiating the communication. The host can be a PC or another embedded system. With this relationship, the bootloader developer will need to consider the effort needed for developing and testing the Slave software interface.

In the LPC2000 family supports an on chip UART bootloader. Although this utility has the ability to use the UART channel it is NOT support. FlashMagic provides support for the UART channel. The Ethernet host software will be discussed further in section 4.

Remark: FlashMagic can be downloaded for free at:

/redirect /

2.2 Bootloader exit

Upon completion or timeout, the exit strategy is usually a reset or power cycle with the application removing the Entry condition (before next power on). It is also possible to issue the ISP “GO” command, allowing the controller to be redirected to the user application.

3. Target design

3.1 Prerequisites

It is assumed that the microcontroller has an enabled Ethernet block found in the LPC23xx/24xx series.

The assigned Media Access Controller (MAC) Address on the controller must be a unique address that **must** not be duplicated on the same Layer 2 broadcast collision domain.

Remark: When connecting directly from a workstation to the microcontroller a crossover Ethernet cable may be needed.

Only one node may establish communications with the microcontroller at any given time.

3.2 Communications design

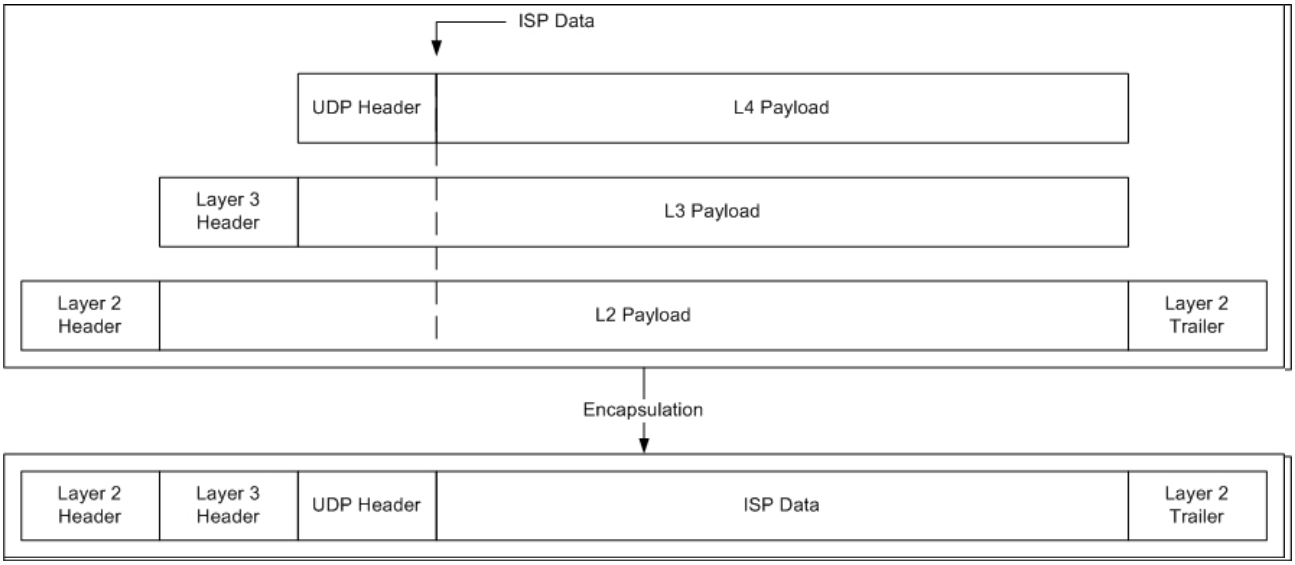
The ISP/IAP communications protocol was originally designed to be used with the UART serial interface and already handles the “handshaking" procedure. To keep the overhead on the embedded system to a minimum the Ethernet will function as simple means of setting packet filters and transporting data without any synchronization.

To mimic an initial synchronization process with the microcontroller, the Host will send a "**?**<CR><LF>" to the microcontroller and wait for a "**Synchronized**<CR><LF>" to be sent back. As the Host receives the "**Synchronized**<CR><LF>" string, it will also reply with "**Synchronized**<CR><LF>" and wait for an "**OK**<CR><LF>". After the synchronization has been established, the host will send the unlock code "**U 23130**<CR><LF>".

Remark: In case of a communication error, the microcontroller has to be reset and the synchronization and unlock procedure have to be repeated.

3.2.1 User Datagram Protocol (UDP)

The UDP packet itself is a "connectionless" oriented protocol in which the transmitting node doesn't care if the packet arrives at its destination. UDP has the advantage that it doesn't need to maintain an active "handshaking" session to transfer data, making it more suitable on systems with limited resources.



(1) Depiction of the combined Layers 2, 3, and 4

Fig 1. Data encapsulation

The Ethernet bootloader accepts data based upon Layer 2 frames by checking the destination address of each frame the Ethernet controller receives. To simply the development process on the windows platform the .NET framework is used to transmit UDP packets. UDP packets however are encapsulated within the Layer 2 frame, meaning that the actual data needed for the Ethernet bootloader within each frame is offset.

For the microcontroller to send data back to the host, it needs to swap the destination and source fields from the Layer 2, 3, and 4 headers. Additionally, it needs to perform a checksum calculation with the data returned from the ISP handler.

3.2.2 Filters

After the microcontroller received the synchronization packet ("**?**"), it will only respond to packets that have the identical source MAC address, UDP source port and UDP

destination port. This feature is implemented for additional protection to prevent stray packets from interfering in the communication session.

3.2.3 EMAC descriptors

A typical windows workstation will generally be able to transmit data at a much higher rate than an embedded system. Since the Ethernet packets are “connectionless”, the Ethernet controller doesn’t care if it lost a packet or not. However, the ISP/IAP protocol requires the host send 20 UUEncoded data packets without receiving any acknowledgment until after the 20th packet. In order to prevent the loss of data packets due to the limited buffer, the EMAC descriptors have been modified.

- The descriptor size has been modified to only accept up to 120 bytes of Layer 2 Ethernet data.
- The number of Rx descriptors has been increased from 4 to 25 to buffer the 20 UUEncoded data packets.
- The number of Tx descriptors has been decreased from 4 to 3 to accommodate the additional Rx descriptors.

In typical applications the Layer 2 payload is around 1500 bytes, thus the user must ensure that no other node is sending anything to that particular microcontroller.

Remark: LPC23xx/24xx Rev - will not work properly using the given EMAC descriptor settings. See the “Ethernet.3” in the LPC23xx/24xx errata sheet.

3.2.4 Checking for incoming data

In order for the bootloader to check if a packet has arrived, it polls a circular receive buffer and checks to see if the number of packets “consumed” matches the number of packets “produced”. The Ethernet hardware increments the MAC_RXPRODUCEINDEX and the Ethernet driver will increment the MAC_RXCONSUMEINDEX. While the two values match, the bootloader waits for a packet.

Remark: When the “MAC_RXCONSUMEINDEX” and “MAC_RXPRODUCEINDEX” indexes match, the receive buffer is empty.

3.3 Modifying the bootloader

3.3.1 Setup File (sbl_config.h)

This file configures the secondary bootloader. The user should change this according to their application.

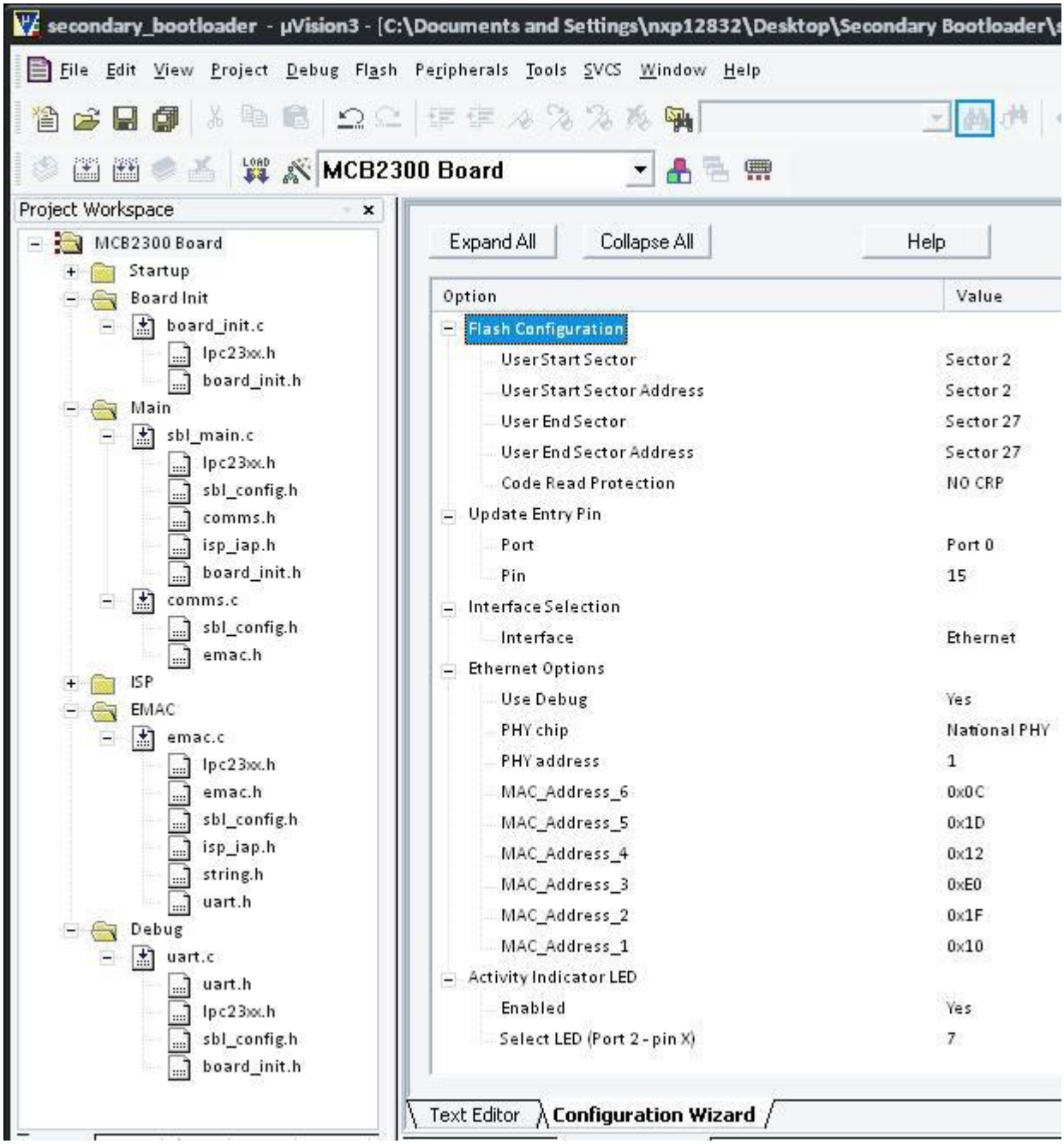


Fig 2. Keil's uVision Configuration Wizard for the sbl_config.h header

Table 1. Configuration parameters for sbl_config.h

Configuration Parameter Definitions	
Flash Configuration	
USER_SECTOR_START	Flash sector that contains the user code entry point
USER_START_SECTOR_ADDRESS	Address of USER_SECTOR_START
USER_END_SECTOR	Last flash sector that contain user code
USER_END_SECTOR_ADDRESS	Address of USER_END_SECTOR
CRP (Code Read Protection)	Sets the Code protection value (CRP1,2,3)
Update Entry Pin	
ISP_ENTRY_GPIO_REG (Port)	Port address used by the Entry Pin
ISP_ENTRY_PIN (Pin)	Pin number used by the Entry Pin
Interface Selection	
USE_ETHERNET (Interface)	Selects interface method, “Other” disables Ethernet
Ethernet Options	
ETHERNET_DEBUG (Use Debug)	Enables or Disables UART0 debug output
MYMAC_6 (MAC_6)	1st Byte of the MAC Address
MYMAC_5 (MAC_5)	2nd Byte of the MAC Address
MYMAC_4 (MAC_4)	3rd Byte of the MAC Address
MYMAC_3 (MAC_3)	4th Byte of the MAC Address
MYMAC_2 (MAC_2)	5th Byte of the MAC Address
MYMAC_1 (MAC_1)	6th Byte of the MAC Address
Activity Indictor LED	
LED_ENABLED (Enabled)	Enabled or Disables Status LED
PORT2_PIN (Select LED)	Select the LED pin of Port 2 (0-7)

Remark: The MAC Address follows the following format:

“MAC_6 - MAC_5 - MAC_4 - MAC_3 - MAC_2 - MAC_1”

3.3.2 Entry pin

This specifies which pin is used to detect whether or not the Ethernet bootloader should start after reset.

3.3.3 Activity indicator LED

When the Activity Indicator LED is blinking it indicates that the Ethernet bootloader is currently running in ISP mode.