Intel® IXP42X Product Line and IXC1100 Control Plane Processor: I²C Implementation Using the GPIO Pins

Application Note

July 2004

Document Number: 252137-004

Contents

Figures

Tables

Revision History

Date	Revision	Description
July 2004	004	Updated Intel® product branding.
August 2003	003	Updated Section 4.1.
June 2003	002	Modified document for inclusion of Intel® IXP4XX Product Line of network processors.
November 2002	001	Initial release of document. (Published as *Intel® IXC1100 Control Plane Processor—I2C Implementation Using the GPIO Port Application Note*.)

1.0 Introduction

1.1 Scope

This application note describes the implementation of the I²C (Inter-IC) using the Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor GPIO pins. This includes the I²C protocol, hardware interface implementation, and I²C software written for the Intel® IXP42X product line and IXC1100 control plane processors running under Wind River Systems* VxWorks*.

Details regarding I²C architecture, performance, and register set are not described in this document. It is assumed that the reader is familiar with the I²C bus protocol.

1.2 Related Documentation

Title	Document Number
Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Developer's Manual	252480
Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Datasheet	252479
Intel® IXP400 Software Programmer's Guide	252539
I²C-Bus Specification from Philips Semiconductors*	(Available at www.philips.com)

1.3 Acronyms

EEPROM	Electrically Erasable Programmable Read-Only Memory
GPIO	General-Purpose Input/Output
I²C	Inter-IC
IC	Integrated Circuit
MSB	Most-Significant Bit
SCL	Serial Clock
SDA	Serial Data
SoC	System-on-Chip

2.0 Overview

The Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor are a highly integrated System-on-Chip designed to provide greater design flexibility and reduce system-development costs. The Intel® IXP42X product line and IXC1100 control plane processors feature many integrated features that simplify system design requirements in embedded applications. These on-chip features include a PC133 SDRAM controller, interrupt controller, GPIO interface, PCI controller, UARTs, watchdog timer (WDT), general-purpose timers, and three Network Processor Engines (NPEs) for the Ethernet and UTOPIA interface.

While the Intel® IXP42X product line and IXC1100 control plane processors do not have dedicated on-chip I^2C support hardware, this I^2C bus can be simulated by software control of two GPIO pins.

This application note provides an overview of the I^2C bus interface and the EEPROM, as well as details on an application interfacing two GPIO pins (I^2C master) to a two-wire EEPROM (I^2C slave).

2.1 I^2C Bus Overview

The I^2C bus is a serial bus developed by Philips Electronics* and a two-wire, bidirectional serial bus that provides a serial data (SDA) line and a serial clock (SCL) line interface to exchange information between devices. The bus is a multiple-master bus including arbitration and collision detection mechanic to prevent data corruption if multiple devices attempt to control the bus at the same time.

Each device on the I^2C bus has its own unique address and can operate as a transmitter or receiver. A *master* is defined as a device that initiates, controls, and terminates a transfer. A *slave* is defined as any device addressed by a master.

Data is transmitted to, and received from, the I^2C bus via a buffered interface. Control and status information is relayed through a set of memory-mapped registers. (For a complete list of I^2C bus features, capabilities, and operation details, see the *I^2C-Bus Specification*.)

2.2 I^2C EEPROM Overview

Serial EEPROM devices are popular for storing system configuration parameters, user settings, measurement data, and other non-volatile information. In a typical usage model, an SoC processor acts as a master device that controls the exchange of data with the EEPROM.

All I^2C bus-compatible devices are incorporated on-chip to enable them to communicate directly with each other via the I^2C bus.

3.0 I^2C Interface

I^2C can be implemented using two GPIO pins of the Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor, including the I^2C standard communication protocol, hardware interface implementation, and I^2C software written for the Intel® IXP42X product line and IXC1100 control plane processors running under VxWorks.

3.1 Functional Description

The I^2C bus defines a serial protocol for passing information between agents on the I^2C bus and uses a two-wire pin interface consisting of an SDA line and an SCL line. Each device on the I^2C bus is recognized by a unique 7-bit address and can operate as a transmitter or receiver.

The I^2C bus serial operation uses an open-drain, wired-AND bus structure. This enables multiple devices to drive the bus lines and to communicate status about events such as arbitration, wait states, and error conditions.

For example, when a master drives the SCL during a data transfer, it transfers a bit every time the clock is high. When the slave is unable to accept or drive data at the rate that the master is requesting, the slave can hold SCL low — between the high states — to insert a wait interval. The master's clock can only be altered by a slow slave peripheral keeping the clock line low or by another master during arbitration.

The I^2C bus allows design of a multi-master system, meaning more than one device can initiate data transfers at the same time. To support this feature, the I^2C bus arbitration relies on the wired-AND connection of all I^2C interfaces to the I^2C bus.

Two masters can drive the bus simultaneously, provided they are driving identical data. The first master to drive SDA high — while another master drives SDA low — loses the arbitration. The SCL consists of a synchronized combination of clocks generated by the masters using the wired-AND connection to the SCL.

Table 1 summarizes I^2C interface signals.

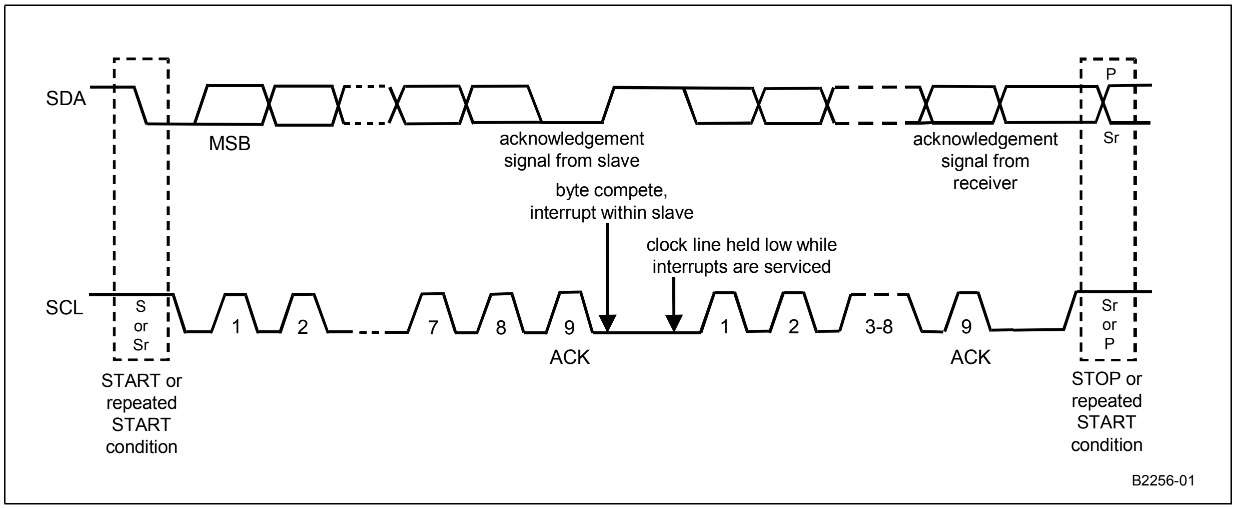
Table 1. Signal Descriptions

Signal	I/O	Description
SDA	Bidirectional	Serial Data
SCL	Bidirectional	Serial Clock

A master is the device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

Each data size is 8 bits long, and the number of data packets that can be transmitted per transfer is unrestricted. Each data packet sent or received is followed by an acknowledge bit. Figure 1 shows the typical I^2C standard communication wave form. The I^2C standard communication protocol consists of control and data signals.

Figure 1. I^{2}C Standard Communication Protocol


SDA
SCL
P
MSB
acknowledgement signal from slave
byte compete, interrupt within slave
clock line held low while interrupts are serviced
acknowledgement signal from receiver
Sr
S or Sr
1
2
7
8
9
ACK
1
2
3-8
9
ACK
Sr or P
START or repeated START condition
STOP or repeated START condition
B2256-01

3.2 Control Signals

Control signals start and stop transfers on the I^{2}C bus and send and receive acknowledgments when appropriate. Changes in the data line — while the clock is high — are interpreted as control signals.

3.3 Data Signals

The I^{2}C bus is a two-wire, bidirectional serial bus that provides an SDA line and an SCL line interface to exchange information between devices (transmitter and receiver). A byte is the biggest data *packet* that can be transmitted between acknowledgments.

Data signals differ from control signals in that, during transfers, the data line remains stable whenever the clock signal is high. All data transfers on the bus are Most-Significant Bit (MSB) first.

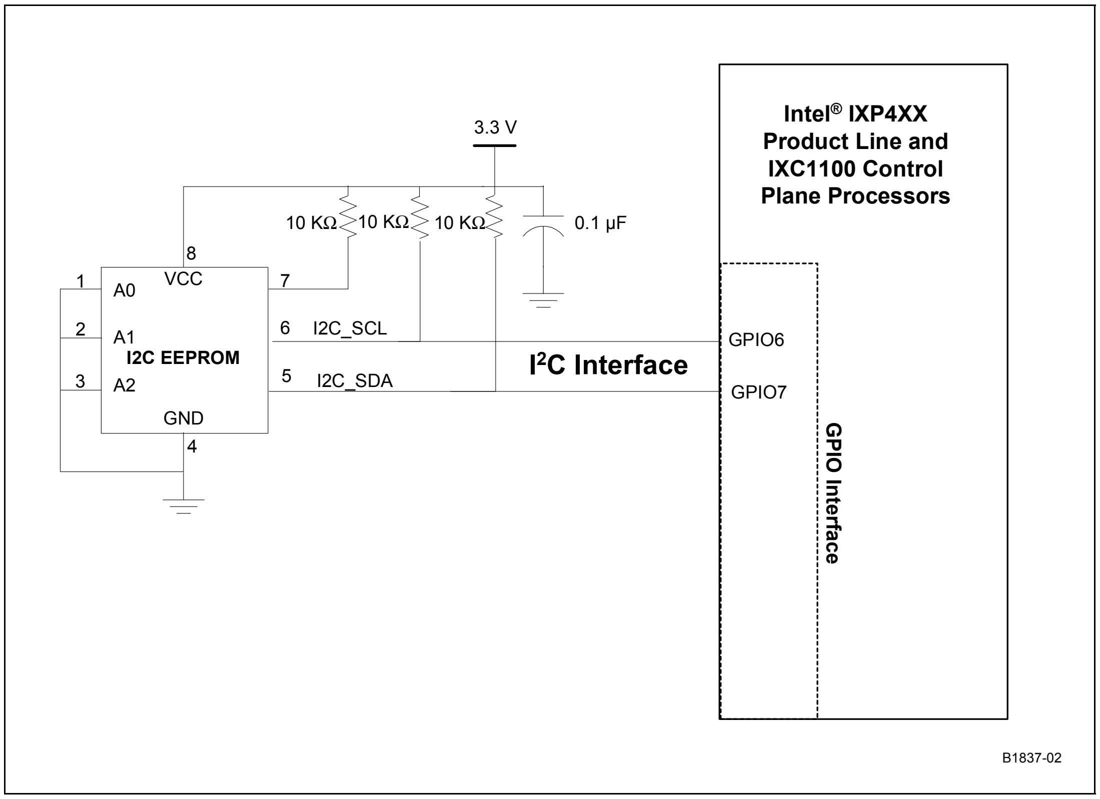
The value of the data line, during a clock pulse, signals the value of that bit on the wire. The value of the next bit is set before the next clock pulse and so on.

3.4 I^{2}C Hardware Interface

Figure 2 shows how to connect the I^{2}C interface to a 256-byte, I^{2}C EEPROM 7-bit addressing mode (Philips PC8582C-2T/03) using two GPIO pins (GPIO6 and GPIO7) of the Intel® IXP42X product line and IXC1100 control plane processors.

For complete details on I^{2}C operation, see the *I^{2}C-Bus Specification*.

Figure 2. I²C EEPROM Interface



All devices connected to the I²C bus — other than the GPIO controller — are automatically considered slaves. As shown in Figure 2, the I²C EEPROM acts as either a slave transmitter or slave receiver.

Because the Intel® IXP42X product line and IXC1100 control plane processors do not have I²C bus interface controller on-chip implementation, the I²C protocol is emulated by two pins of the GPIO controller (GPIO6 & GPIO7). This component depends on functionality provided by the GPIO driver. The GPIO software driver is used to toggle two pins on the GPIO controller, which represent the SDA and SCL lines of an I²C bus.

The official I²C bus protocol supports three modes of transfer rates:

- Standard Mode — Up to 100 Kbps
- Fast Mode — Up to 400 Kbps
- High-Speed Mode — Up to 3.4 Mbps

Only the standard mode is supported in the sample code referenced in Section 4.1, “I²C Software Source Code” on page 12.

4.0 I^{2}C Software Protocol

The I^{2}C software driver is written for the Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor running under VxWorks. An actual I^{2}C bus is emulated using two GPIO pins (GPIO 6 and GPIO 7). One pin is for data signals (SDA), and one pin is for clock signals (SCL).

The software driver enables client software to operate as either a transmitter or receiver, depending on its function. However, only one master device is supported (for example, the Intel® IXP425 Network Processor). This is because the GPIO controller, which is used to drive the SDA and SCL lines, does not have open-connector outputs. Because of this, no other device is able to drive the data like while the processor is driving the data line.

To facilitate understanding of the I^{2}C protocol and the way it is implemented in this driver, these main control and data signals must be understood:

- **START Signal** — Initiates a data transfer request.

  A device can initiate communication by sending a START signal when another device is bus master. A START signal is represented by a HIGH to LOW transition on the SDA line while the SCL line is HIGH.

  If the bus is free (both SDA and SCL are HIGH), the SDA line is forced LOW. If the bus is not free, it remains unchanged.

  If successful, the bus is considered busy after the generation of a START signal and can only be released by the generation of a STOP signal.

- **STOP Signal** — Sends a STOP signal on the bus.

  A STOP signal is represented by a change in the SDA line from LOW to HIGH while the SCL line is HIGH. This function may be called at any time during a data transfer. The bus is considered free after the generation of a STOP signal.

- **AckSend** — Data transfer with acknowledge is obligatory and is generated after each byte is transmitted.

  This function generates an acknowledge signal on the I^{2}C bus. It is used by the Intel® IXP42X product line and IXC1100 control plane processors when it is running as master receiver. It generates an acknowledge related clock pulse, during which the SDA line is pulled LOW to indicate an acknowledgment to the slave transmitter. A master receiver can signal the end of data to a slave by not generating an acknowledge on the last byte that was clocked out of the slave.

- **AckReceive** — Receives an acknowledge signal on the I^{2}C bus.

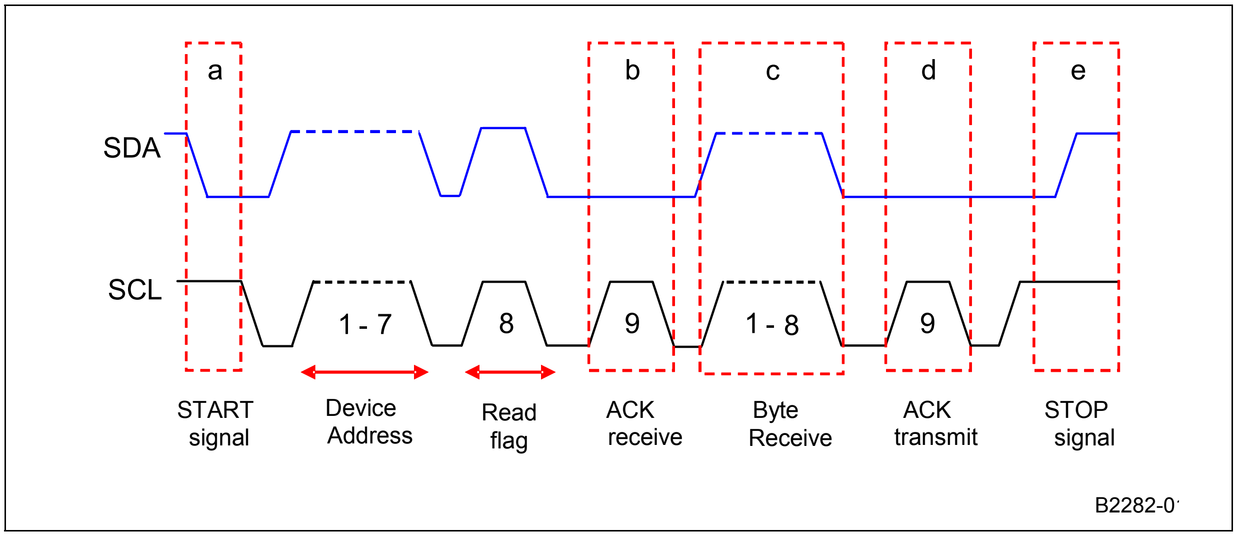
  This signal is used by the Intel® IXP42X product line and IXC1100 control plane processors when running as master transmitter. An acknowledge related clock pulse is generated, and the value of the SDA line tested to see if the slave receiver has acknowledged. If, for some reason, the receiver does not acknowledge within the time allowed, the transmitter can generate a STOP signal to terminate the transfer.

- **ByteTransmit** — Transmits a byte on the I^{2}C bus.

  In this case, the Intel® IXP42X product line and IXC1100 control plane processors are acting as master transmitter. All data transfers on the bus are MSB (Most-Significant Bit) first. The value of the data line during a clock pulse signals the value of that bit on the wire. The value of the next bit is set before the next clock pulse, and so on. Figure 4 shows where the value 0xC0 is transmitted.

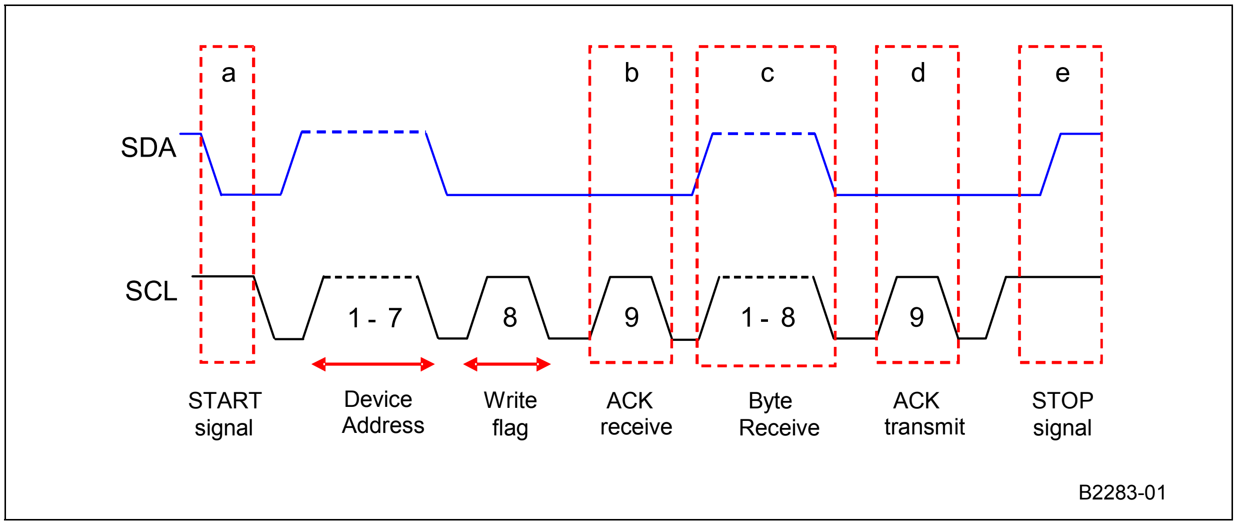
- **ByteReceive** — Analogous to the transmit function except in this case the SDA line is toggled by the slave transmitter (for example, the Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor acting as master receiver). Again, all byte transfers are MSB first.
- **ReadTransfer** — Reads "num" bytes from the slave device addressed by "devAddr" at offset "offset." This function encapsulates a number of the primitive protocol functions described previously, as can be seen in Figure 3. Sections "c" and "d" would occur "num" times, once for the offset and several times for each byte that is to be transmitted.

Figure 3. Complete Read Transfer Sequence



- **WriteTransfer** — Writes "num" bytes to the slave device addressed by "devAddr" at offset "offset." It encapsulates a number of the primitive protocol functions described previously as seen in Figure 4. Sections "c" and "d" would occur "num" times, once for the offset and several times for each byte that is to be transmitted.

Figure 4. Complete Write Transfer Sequence



4.1 I²C Software Source Code

For example source code listings for the I²C software driver, see the Wind River* (http://www.windriver.com) Intel® IXDP425 / IXCDP1100 Development Platform Board Support Package.