Keywords: MAXQ2000, I2C

REFERENCE SCHEMATIC 4267

Proven Implementations of the I²C Bus

Sep 11, 2008

Abstract: This application note briefly reviews the history of the I²C bus. It then presents I²C configurations proven to ensure easy communication between the master and slaves on the bus. Examples include schematics and code. Appendix 1 contains helpful definitions of terms used in this article.

A similar article was published on the *Embedded.com* website, August 8, 2007.

Introduction

The Inter IC (I^2C) bus was developed by Philips Semiconductor in the early 1980s to simplify electronic products by reducing the number of parallel data lines. Version 1.0 of the I^2C-bus specification, released by Philips® in 1992, defined a simple, 2-wire, bidirectional bus for communication between ICs. By 1998, the I^2C bus had become the de facto standard for low-speed IC-to-IC communications. At that time more than 50 licensed companies were using the standard, and the I^2C interface was included in more than 1000 different ICs.

The I^2C bus configurations presented in this article have been proven to ensure easy communication with slave devices on the bus. Each implementation includes examples in the form of schematics and code. The reader should be familiar with the following documents:

- *The I²C-Bus Specification*, Version 2.1, January 2000.
  Philips Semiconductor document #9398 393 40011.
- *The I²C-bus and how to use it*, April 1995.
  Philips Semiconductor document #98-8080-575-01.
- *System Management Bus (SMBus) Specification*, Version 2.0, August 2000.

Background and Discussion

The I^2C bus can operate in Standard mode, Fast mode, or High-Speed (Hs) mode, with maximum data rates of 100kbps (Standard mode), 400kbps (Fast mode), 1.7Mbps (Hs mode with C_b = 400pF), and 3.4Mbps (Hs mode with C_b = 100pF).

The original Standard mode incorporated 7-bit addressing, which allowed 112 slave addresses. As I^2C-bus systems demanded more slave devices, 10-bit addressing was introduced to allocate more slave addresses.

Fast mode added useful features to the slave devices. The maximum data rate was quadrupled to 400kbps.

Fast-mode I^2C also dropped the support of similar buses often linked with the I^2C bus, i.e., buses that were no longer compatible with the higher data rate. To suppress noise spikes, Fast-mode devices were given Schmitt-triggered inputs. In addition, the SCL and SDA lines of an I^2C-bus slave device were required to exhibit high impedance when power was removed.

Hs mode was created mainly to increase the data rate—up to 36 times faster than Standard mode. For I^2C buses operating in Hs mode, the most significant change affects low-to-high transitions on the SCL line. Because the pullup resistors used in Standard and Fast mode cannot produce rise times fast enough to meet the Hs-mode specification, most Hs-mode systems must include active pullups on the SCL line. Other changes include an Hs-mode compatibility request, issued by the Hs-mode master in Standard or Fast mode using an 8-bit master code. If Not-Acknowledge (a bit name within the I^2C frame) remains high immediately following the master code, then all communications remain in Hs mode until a stop occurs. The waveforms of **Figure 1** illustrate the use of the master code for entering Hs mode.
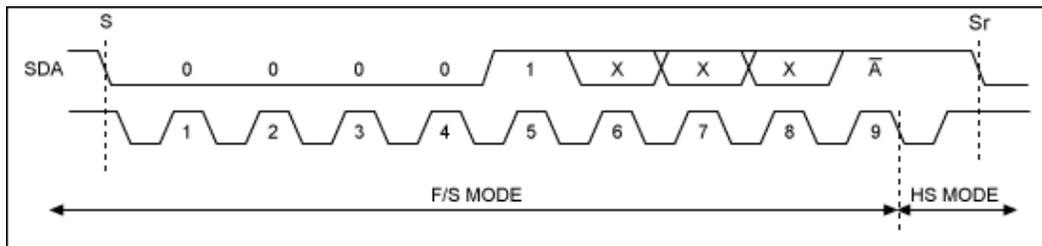

Figure 1. These waveforms represent a transfer from Standard- or Fast-mode I^2C to Hs mode.

Proven Implementations for I^2C On-Chip Peripherals

The PIC18F442 microcontroller (µC) includes an I^2C peripheral that supports Standard- and Fast-mode I^2C. **Figure 2** shows an application circuit using that peripheral to collect samples from a 16-bit serial-output ADC (MAX1169). When data is received by the PIC®, it is immediately transmitted at 115kBaud by a firmware UART. The RS-232 transceiver (MAX3232E) then allows the data to be captured by a personal computer's standard serial port. All assembly source files needed to implement Fast-mode I^2C on the PIC's on-chip peripheral are contained in the file I2C_on_chip_asm.zip. A tool called MPLAB IDE Version 6.10.00 was used to develop assembly code for the Figure 2 circuit.

Figure 2. The I^{2}C peripheral internal to this PIC18F442 µC connects to a 16-bit ADC (MAX1169).

Proven Implementations for Bit-Banging I2C

The MAXQ2000 is a low-power 16-bit RISC µC capable of bit-banging Standard-mode, Fast-mode, and even 1.7MHz Hs-mode I^{2}C signals. **Figure 3** shows a general-purpose schematic of the MAXQ2000 operating as an I^{2}C master in all three modes. An active pullup is included in the MAXQ2000 to support the 1.7MHz Hs mode. The file I2C_bit_bang_asm.zip contains all assembly source files needed to bit-bang Standard- and Fast-mode I^{2}C systems using the µC's GPIO lines.
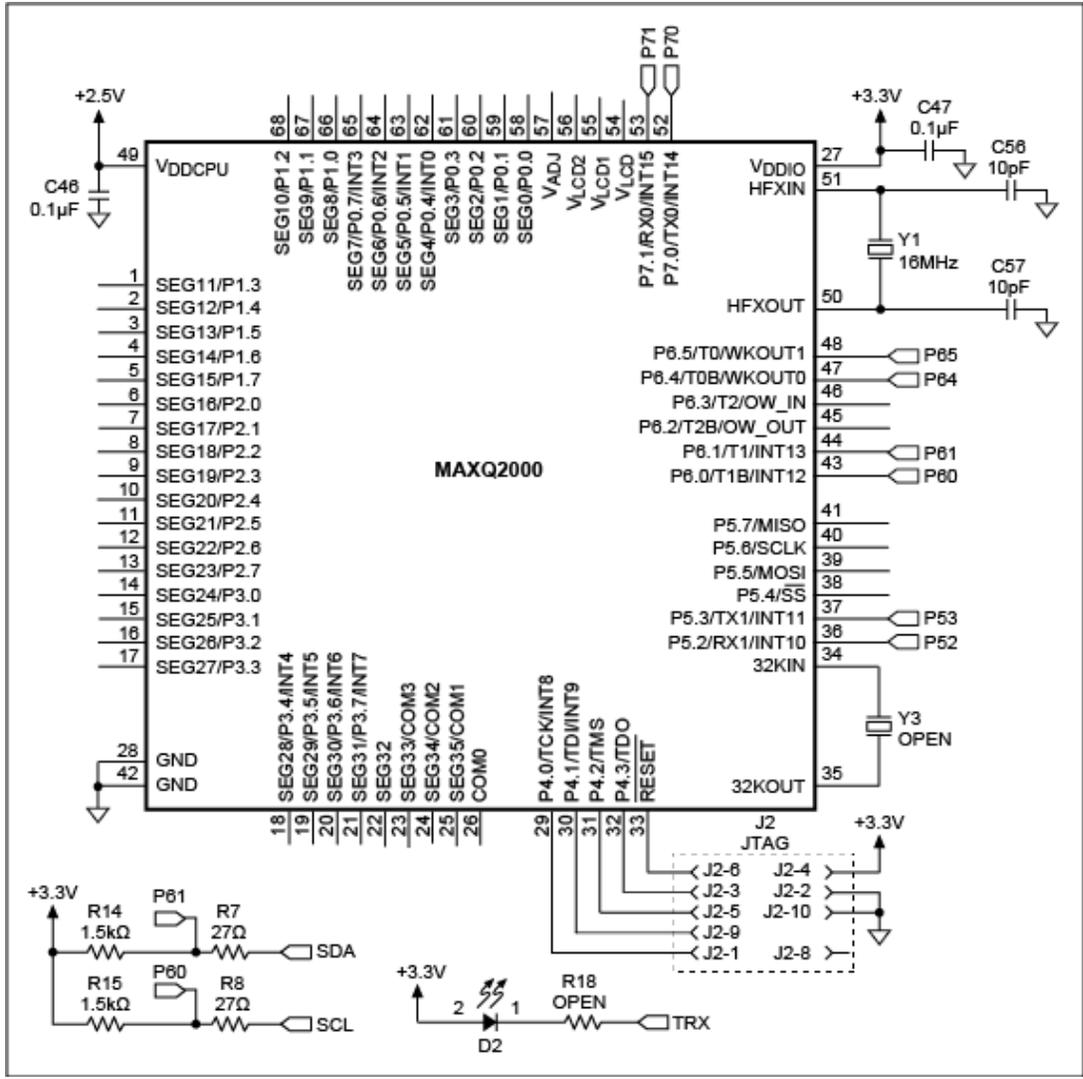
Figure 3. This schematic (based on the MAXQ2000 low-power LCD μC) depicts an I²C master capable of operating in Standard, Fast, or Hs mode.

The default clock speed for the MAXQ2000's I²C firmware is 100kHz, but the μC's 16MHz system clock allows a bit-banged I²C interface to run as fast as 400kHz. The following code example shows how to capture interrupt events and handle them in a simple interrupt-service routine. The development tool used is MAX-IDE Version 1.0 (build date: Nov 18 2004), which can be downloaded for free from the Maxim website. The example software for this application note is bundled into a single file available for download.
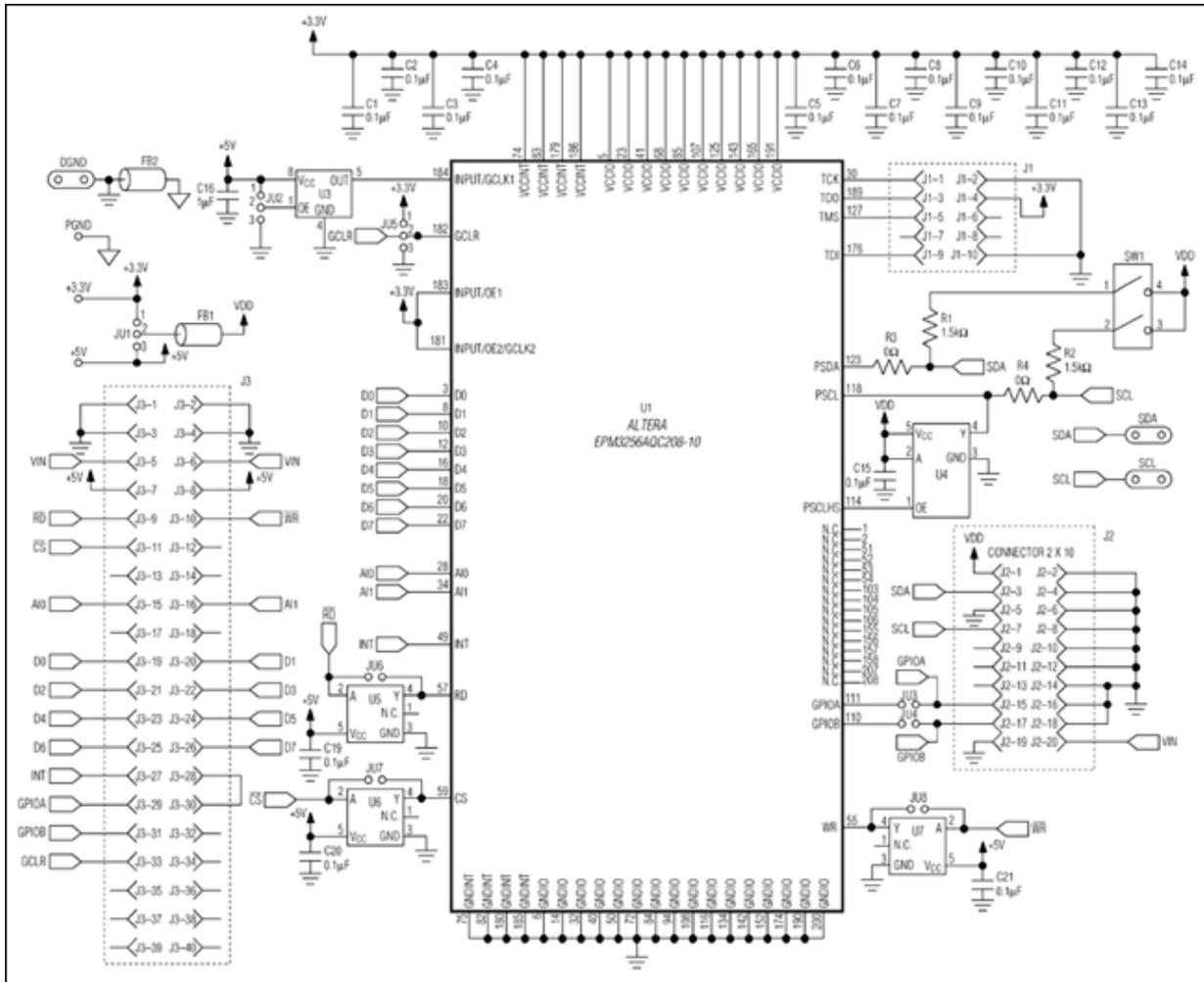
The file I2C_bit_bang_c.zip contains all the C-source files required to bit-bang a Standard- or Fast-mode I²C link using the MAXQ2000's GPIO lines. This example in Figure 3 is based on a maxqi2c library consisting of two files, maxqi2c.h and maxqi2c.c. When added to your project, these files allow 100kHz or 400kHz I²C operation on any GPIO pin. For the C code to produce I²C at the specified speed, however, Y1 in Figure 3 must be a 20MHz crystal instead of a 16MHz crystal. The IAR Embedded Workbench® IDE for the MAXQ2000 Version 1.12B (FAE edition) was the development tool used in this case. You can register and download a free copy of the IAR Embedded Workbench for the MAXQ2000 (4K Kickstart Edition) at the IAR website.

The file HSI2C_bit_bang_asm_c.zip contains all the assembly and C source files required to bit-bang a 1.7MHz Hs-mode I^{2}C link using the MAXQ2000 GPIO lines. By mixing assembly and C code, the source code in this example takes full advantage of the strength of each code type. Assembly optimizes speed, and C accomplishes many things in just a few lines of code. The timing-critical Hs-mode I^{2}C receive function (hsi2cRecv) is defined in the assembly source file hsi2c.asm.

The main C source file initializes the MAXQ2000's UART at 115.2kBaud. For the source code to produce a 1.7MHz Hs-mode I^{2}C link, Y1 in Figure 3 must be a 20MHz crystal. The main C source file calls the hsi2cRecv function when needed, and a printf function transfers data from the on-chip UART, which is set for the 8-N-1 data format. The Rowley CrossWorks IDE Version 1.0 (Build 2 licensed copy) was the development tool used in this example. To obtain the CrossWorks IDE, contact Rowley Associates Limited or visit their website.

Proven Implementations for I^{2}C IP Core Peripherals

An I^{2}C IP core called DI2CM (by Digital Core Design) is used in many CPLD and FPGA devices. The DI2CM IP core converts a parallel interface to an I^{2}C interface, and implements a true Hs-mode master capable of clock speeds as high as 3.4MHz. In the application circuit of **Figure 4**, the Altera® EPM3256AQC208-10 CPLD includes a DI2CM IP core. This circuit requires a 40MHz crystal oscillator (U3) to achieve compatibility with both 1.7MHz and 3.4MHz Hs-mode I^{2}C. A 3-state logic buffer with output enable allows the DI2CM IP core to produce an active pullup on the SCL line, as is required by Hs-mode I^{2}C. Logic inverters are provided at U5, U6, and U7 to support both an active-high and an active-low memory-mapped parallel interface.

(PDF, 841KB)

Figure 4. Configured as shown, the DI2CM IP core in this Altera EPM3256AQC208-10 CPLD implements an I^{2}C Hs-mode master.

The file HSI2C_IP_core_asm.zip contains all the assembly source files needed to implement an Hs-mode I^{2}C master, obtained by interfacing the 68HC16's memory-mapped parallel interface to an Altera EPM3256 CPLD programmed with the DI2CM IP core. The development tool used in this example was the Motorola® 68HC Macro Assembler Version 4.1.

Conclusion

Because the I^{2}C bus is currently the industry's most widely used serial bus, it behooves a system designer to have a handful of proven implementations readily available. The method that you choose—on-chip, bit-banged, or IP-core implementation—depends principally on the system processor. Nonetheless, nothing is easier than using an approach that is proven and already works. This article includes a working reference for each of the three methods.

Appendix 1: Definition of Terms

- CPLD: Complex programmable logic device
- DI2CM: An Hs-mode I^2C IP core from Digital Core Design
- FPGA: Field-programmable gate array
- GPIO: General-purpose input outputs
- hsi2cRecv: Hs-mode I^2C receive function, written in assembly language
- Hs-mode: High-speed I^2C mode
- IC: Integrated circuit
- IDE: Integrated development environment
- IP: Intellectual property
- I^2C: Inter-IC bus
- MAXQ2000: A low-power 16-bit RISC microcontroller
- RS-232: Recommended standard #232
- SCL: Hardware serial-clock line for the I^2C bus
- SDA: Hardware serial data line for the I^2C bus
- UART: Universal asynchronous receiver-transmitter
- µC: Microcontroller
- 8-N-1: 8 bits, no parity, and 1 stop bit

References

1. Philips Semiconductor Staff, "The I^2C-Bus Specification," Version 2.1, January 2000. Philips Semiconductor document #9398 393 40011.
2. Philips Semiconductor Staff, "The I^2C-bus and how to use it," April 1995. Philips Semiconductor document #98-8080-575-01.
3. SBS Implementers Forum, "System Management Bus (SMBus) Specification," Version 2.0, August 2000.
4. Ted Salazar, in Maxim application note 2394, "Interfacing the MAX1169 ADC to a PIC Microcontroller."
5. Ted Salazar, in Maxim application note 3568, "Using the MAXQ2000 with the MAX4397 to Create an Inexpensive Audio/Video Source Selector."
6. Maxim application note 3588, "Software I^2C Driver for the MAXQ2000 Microcontroller."
7. Maxim application note 3561, "Getting Started with MAX-IDE."
8. Digital Core Design Staff, "DI2CM I^2C Bus Controller Master, HDL Core Specification," Version 2.25.ma.

Altera is a registered trademark and registered service mark of Altera Corporation.
IAR Embedded Workbench is a registered trademark of IAR Systems AB.
Motorola is a registered trademark and registered service mark of Motorola Trademark Holdings, LLC.
Philips is a registered trademark of Koninklijke Philips Electronics N.V. Ltd.
PIC is a registered trademark of Microchip Technology Incorporated.

Related Parts		
MAX1169	58.6ksps, 16-Bit, 2-Wire Serial ADC in a 14-Pin TSSOP	Free Samples
MAXQ2000	Low-Power LCD Microcontroller	Free Samples

More Information

For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 4267: http://www.maximintegrated.com/an4267
REFERENCE SCHEMATIC 4267, AN4267, AN 4267, APP4267, Appnote4267, Appnote 4267
© 2013 Maxim Integrated Products, Inc.
Additional Legal Notices: http://www.maximintegrated.com/legal