AVR325: High-speed Interface to Host EPP Parallel Port

8-bit AVR® RISC Microcontroller

Application Note

Features

- **Fast Bi-directional Point-to-Point Data Transfer with Intel® PC Host**
- **Code supports all AVR® Devices with 12 I/O Pins or more**
- **Interrupt Driven**
- **No Additional Hardware Required**
- **Transfer 60 Kilobytes/second at 4 MHz Operation**
- **Less than 100 Words of Code Space, No Dedicated RAM Space**
- **Includes Host Program In C Demonstrating use of EPP**

Introduction

This application note describes a method for high-speed bi-directional data transfer between an AVR microcontroller and an off-the-shelf IBM® PC-compatible desktop computer. The interface provides an 8-bit parallel data path, yielding data transfer rates up to 60 kilobytes/second with an AVR processor operating at 4 MHz. This is an order of magnitude faster than a standard RS-232 connection while not requiring complex external interface hardware (like USB and SCSI).

This application note provides the code for both the host and AVR needed to interface an AVR to a host parallel port. It also contains construction details for the required connecting cable and information on host setup.

Background

Since the introduction of the original IBM Personal Computer in 1981 Intel x86 desktop computers have become widespread, fairly well standardized and documented. They offer fast, inexpensive high-capacity data storage, sophisticated user interfaces, and sophisticated modeling and software development tools. However, most methods for connecting external devices to these systems are either relatively slow or require complex software and specialized external hardware.

The Intel x86 PC's parallel I/O port is a compromise between achiving high speed connection and low cost. This port was originally designed as a limited-use one-way interface for driving printers. However, current host chipsets for the Parallel Port support multiple modes of operation to reflect this port's expanded role as a disk and tape interface.

Rev. 2506A–AVR–02/02

One of these enhanced modes of operation, known as EPP – Enhanced Parallel Port – can provide 8-bit bi-directional data transfer at speeds exceeding 1 megabyte/second. Current AVR's do not operate at speeds which allow them to take full advantage of EPP transfer speed, but even a 4 MHz, AVR can achieve a 60 kilobyte/second transfer rate using this protocol. This is fast enough to allow an AVR to perform high-speed peripheral functions, or to allow the host PC to be used as an AVR user interface and storage unit for high-speed data logging.

EPP Protocol Summary

The IBM PC-compatible architecture implements the printer port as a range of addresses, accessed as 8-bit "registers" by the i86 processor's IN and OUT instructions. The original printer port – SPP – was implemented as three registers: data, status, and control; EPP mode expands this to five, and in some cases, eight.

Since a specific parallel port may exist at one of several addresses, the registers are usually referred to by offsets from a common "base" I/O address, for example "base+1" for the Status register. The SPP definition provides for three base addresses, but EPP only supports two of these: 0x378 and 0x278.

The EPP protocol supports reads and writes of two different types referred to as Address and Data over a single 8-bit data bus. Reads and writes of each type are distinguished by a single control signal – nWrite – and Address and Data transfers by two control signals – nDataStrobe and nAddressStrobe. The code provided with this application note implements Data-type transfers only, but the addition of Address-type transfers should be straightforward.

In the PC host, control bits may be set in the EPP Control Register – base+2 – and status bits are returned in the EPP Status Register – base+1. The Control Register is not used for manipulating control lines during a transaction, however – these control lines are driven by the underlying EPP hardware.

The PC host reads or writes Address information by reading or writing the Address Register – base+3 – and Data by reading or writing the Data Register – base+4.

The EPP is fundamentally a master/slave protocol, in that the host is always the initiator of writes and reads over the bus. However, the interrupt signal – nInterrupt – or one of the "spare" signal lines may be used by the peripheral to request service from the host. The supplied code uses nInterrupt – nINTR – for this purpose.

Each Operation Proceeds in Four Steps

1. The host asserts the read/write line – nWrite – and the 8 data lines and then asserts the Data Strobe – nDataStrobe.
2. The peripheral captures (Write) or asserts (Read) the 8 data lines, then asserts the wait line – nWait.
3. The host captures the 8 data lines and then de-asserts the Data Strobe – nDataStrobe.
4. The peripheral de-asserts the wait line – nWait.

Once step 4 is complete, the next operation may proceed from step 1 for EPP 1.9; for EPP 1.7, the next operation may begin after the completion of step 3.

A 10 µs time-out is applied on the operation from the beginning of step 1 to the completion of step 2.

On top of this ISO layer-1 protocol, any of a number of possible layer-2 protocols may be implemented.

For more information on EPP, consult the references listed at the end of this application note.

Implementation

The hostepp.c code was written for Borland's DOS®-based TurboC v2.01compiler; a copy of the TurboC compiler may be downloaded from the URL provided in the References section. AVREPP.ASM was coded using Atmel's AVRASM assembler.

AVREPP supports Data reads and writes. It consumes 12 I/O lines (8 data + 4 control, including 1 external interrupt – INT0), and 7 registers (6 high, 1 low). RAM usage is limited to use of the call stack.

The package consists of an Interrupt Service Routine – ISR – and a simple library containing an initialization routine (ppinit) and two byte-transfer functions (ppgetc/ppputc). These are coded to be stand-alone, suitable for wholesale insertion into another application. (Register definitions and port assignments should be checked for compatibility with the containing application.)

AVREPP.ASM contains a main loop which simply echoes data sent from the host back to the host, adding line feeds when a carriage return is seen.

Buffering

A one-element queue is provided for each direction (read/write). When the host performs an EPP write, the data is stored in the queue until the AVR application retrieves it with ppgetc. The application makes data available to the host by calling ppputc. This function stores the byte in the queue and then asserts the interrupt line – nINTR – notifying the host that data is available. When the host performs a read in response to this signal the queued data byte is provided.

If the host issues an unsolicited Read (e.g., nINTR has not been asserted) there is no data in the read buffer and the request is ignored. This is seen at the host as a Read Time-out.

If the host issues a Write when the write queue is full, an overrun has occurred. EPP doesn't really provide a write flow-control mechanism which would allow the host to avoid overruns – this needs to be dealt with in the ISO layer-2 protocol in the application software. The code supplied with AVREPP simply accepts the new byte, overwriting the previously queued byte and records the event; ppgetc then returns a "!" character at the point in the data stream where the lost byte(s) would have appeared.

Interrupts

The AVREPP ISR is triggered by the EPP Data Strobe signal – nDATASTR – which is connected to the AVR INT0 external interrupt pin. A single EPP transaction (read or write) is processed in two phases, which may result in either one or two AVR interrupts, depending on the implementation of the host EPP hardware.

The first phase is triggered by the assertion of nDATASTR (LOW).This triggers a falling-edge interrupt for INT0. In response to this condition, the ISR first reverses the sense of INT0 to rising-edge, then performs the step-2 operations fetching/asserting the data and asserting nWAIT.

The second phase is triggered by the de-assertion of nDATASTR, which triggers a rising-edge interrupt for INT0. In response to this condition, the ISR reverses the sense of INT0 back to falling-edge for the next transaction, then performs the step-4 operations de-asserting nWAIT.

This two-phase operation is required since the EPP protocol allows the host to take up to a full second to de-assert nDATASTR. To allow for this possibility the AVREPP ISR returns from the interrupt and is re-entered later when nDATASTR is finally de-asserted.

The AVREPP ISR has a heuristic in the form of a test at the end of the first phase to see if the host has already completed step 3; if so, the ISR jumps directly to phase 2 without returning from the ISR, and the transaction completes with only one AVR interrupt.

This allows the ISR to function correctly also when nDATASTR is de-asserted very quickly after the peripheral assert nWAIT.

Host Setup

In order to use the EPP protocol, the I/O chip set must be configured properly using the host's BIOS Setup software at host boot time. The method for configuring the host's Parallel Port depends on the BIOS and chip set used, but it will typically be found in a BIOS Setup section labelled "Peripheral" or "Peripheral Setup". For some systems this may be a sub-section under an "Advanced" heading.The following options, from a system equipped with an SMCFDC37C666IR I/O chipset and an AMI BIOS are typical:

Setup

\Peripheral

\On Board Parallel Port	0378 (port address)
Parallel Port Mode	EPP
Parallel Port IRQ	7

Note that the definition of PPORT in hostepp.c must match the port address specified for the Parallel Port.

Cable Pinout/STK200 Connections

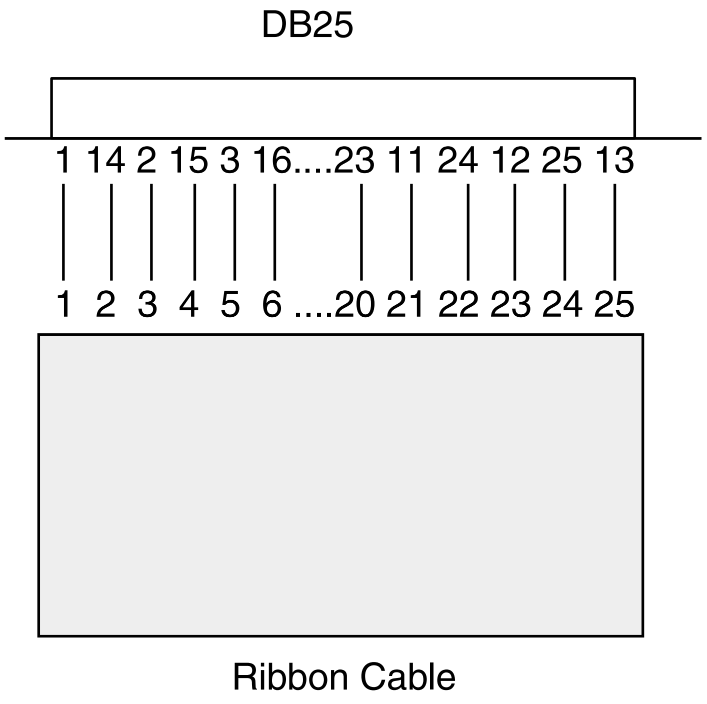
The parallel port connection on an IBM-compatible PC is normally brought out through a female DB25 connector. When the port is configured to operate in EPP mode, the DB25 pins are used as shown in Table 1:

Table 1. Pin Mapping

DB25	AVREPP Pin Use	AVREPP Label	EPP Signal
1	PD4	nWRITE	nWrite
2-9	PC0-7		Data0-7 respectively (Data0 is low)
10	PD5	nINT	nInterrupt
11	PD6	nWAIT	nWait
12-13	--		(spare)
14	PD2	nDATASTR	nDataStrobe
15	--		(spare)
16	--	not used	nReset
17	--	not used	nAddressStrobe
18-25	Gnd		Gnd

When using a ribbon cable, keep in mind that most ribbon cable DB25 plugs connect the DB25 pins to the cable in the order shown in Figure 1.

Figure 1. Parallel Port to Ribbon Cable Connection



EPP 1.7 vs. 1.9

The original EPP implementation EPP v1.7 handled the nWait signal slightly differently from the current (v1.9) implementation. The supplied code has been written for and tested with EPP v1.9.

For a detailed description of both implementations, see the sections “EPP 1.9 OPERATION” and “EPP 1.7 OPERATION” in the SMC FDC37C665/666 datasheet PDF file.

References

Parallel Port Complete, by Jan Axelson

[ISBN 096508191-5]   Lakeview Research
http://www.lvr.com/

This book provides a complete description of the Intel x86 PC Parallel Port and currently supported modes of operation. It includes timing diagrams for each mode, circuit diagrams for external peripherals, and sample host code in Visual Basic.

Standard Microsystems Corporation

http://www.smsc.com/
SMC FDC37C665/666 and 93X datasheets, application notes.

National Semiconductor

http://www.national.com/
PC97338 datasheets

VIA Technologies

http://www.viatech.com/
VT82C686A datasheets

EPP Parallel Port

http://web.tiscalinet.it/nick/pinconpar_epp.htm
EPP pinouts and protocol description, with timing charts

TurboC

http://community.borland.com/
Downloadable copy of Borland TurboC Compiler v2.01 for MS-DOS are avalable in the museum section (registration required).

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Memory
Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 436-4270
FAX 1(408) 436-4314

Microcontrollers
Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 436-4270
FAX 1(408) 436-4314

Atmel Nantes
La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

Atmel Colorado Springs
1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Atmel Smart Card ICs
Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Atmel Heilbronn
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

Atmel Colorado Springs
1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com