Developing Plug-and-Play COM Ports using TI Plug-and-Play Controllers

Heinz-Peter Beckemeyer

July 1997

SLLAE02

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safe-guards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Copyright © 1995, Texas Instruments Incorporated

Contents

Figures

1. Introduction

The installation and configuration of a system can be a major problem for the user, as a result of the wide variety of different plug-in boards and computer systems which are in use. The ISA (Industry Standard Architecture) bus is a widely used standard in the PC industry**.** The ISA bus architecture requires the allocation of resources. These include the I/O address space, a memory address space, the DMA channel and the Interrupt channel, for every ISA plug-in card which is used. However, the ISA standard does not provide any definite hard or software controlled mechanism for allocating the card characteristics. The configuration of each card is usually performed with the help of jumpers or DIL switches. Reference must be made to the documentation of the card manufacturer, in order to avoid conflicts with other ISA cards. For the user this configuration can be a time-consuming and frustrating process.

The definition of the Plug-and-Play ISA standard now makes it possible to develop plug-in cards having full ISA bus compatibility, which will be configured completely automatically by the system. This spares the user every kind of adjustment, such as inserting jumpers or setting DIL switches. A system containing only ISA Plug-and-Play cards can thus be recognized and configured completely automatically, without any kind of intervention by the user. In addition to the logic circuitry which is to be found on the Plug-and-Play compatible ISA card, a program is needed which will automatically configure the card in the PC. Microsoft Windows® 95 is the first operating system which integrates “Plug and Play” at operating system level, and thus supports various bus systems, such as PCMCIA, EISA, PCI and the extended ISA Plug-and-Play bus system.

Texas Instruments supplies Plug-and-Play controllers, which allow the development of Plug-and-Play compatible ISA cards. This Applications Report discusses in detail the development of a Plug-and-Play Windows® 95 compatible serial interface (COM port), based on the TL16C550.

A serial interface circuit which is compatible with the TL16C550 will be termed "16550 compatible" in this Application Report. Where the interface circuit is in addition supported by the operating system Windows® 95, then it will be termed "16550 Windows® 95 compatible".

Windows 95 is a registered trademark of Microsoft Corporation

2. Plug-and-Play Logic

The Plug-and-Play controller provides the interface between the ISA bus and the logic of a Plug-and-Play PC plug-in card. Texas Instruments provides various Plug-and-Play controllers. In this Applications Report, details will be given of a circuit proposal using the Plug-and-Play compatible serial interface (UART) TL16PNP550A and the Plug-and-Play controller TL16PNP100A. Figure 1 shows the typical block circuit diagram of the Plug-and-Play logic circuitry, such as is to be found in the TL16PNP100A or TL16PNP550A.

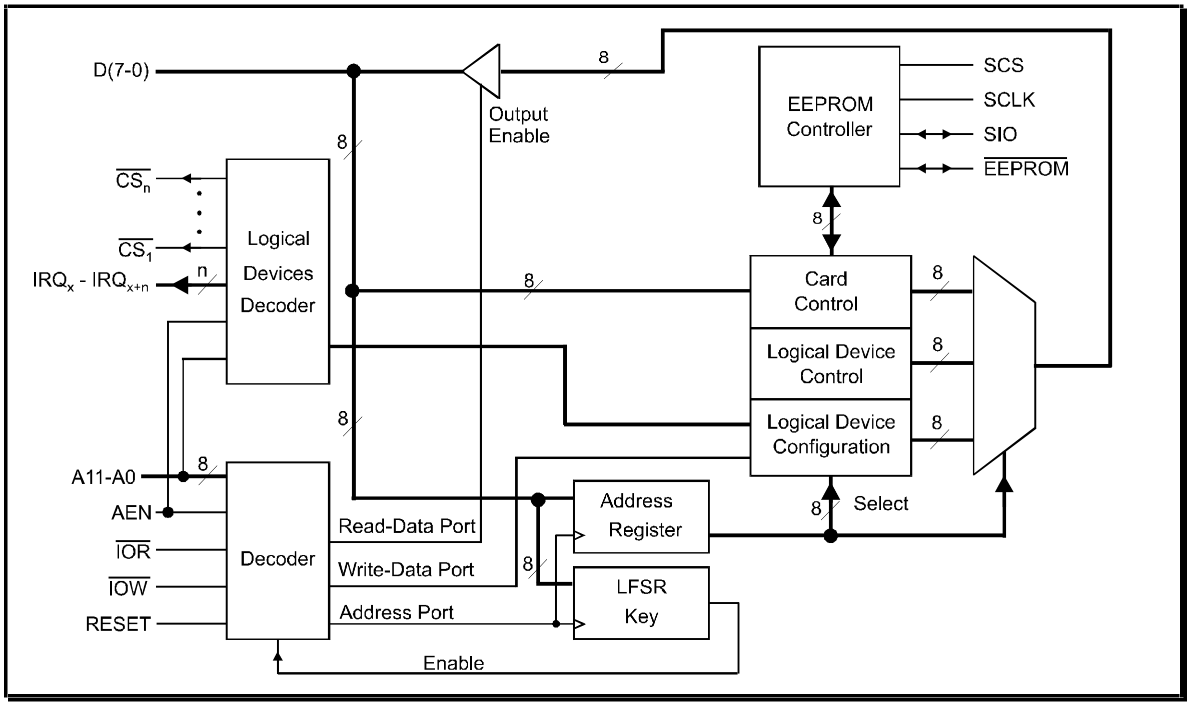

Figure 1: Block circuit diagram of a PnP logic circuit

When starting up, the Plug-and-Play logic is in a quiescent mode, and requires a program in order to be activated and recognized. This recognition is performed by a definite write sequence at the address port. This write sequence is decoded with the aid of an LFSR (Linear Feedback Shift Register). If a valid write sequence is recognized, then the PnP plug-in card will be set into the configuration mode. This is followed by the isolation of each individual Plug-and-Play ISA plug-in card. This isolation procedure is necessary because, when switching on the computer, the PnP cards are at the same I/O address. The isolated card is allocated a card number (Card Select Number), at which the card can be addressed when accessed on a later occasion. After isolation, the card reverts to the configuration mode, and the configuration data of the PnP plug-in card is read out. As soon as all configuration data are known, resources are allocated to the PnP plug-in card.

The PnP plug-in card ultimately leaves the configuration mode, and is switched into an active mode.

Three 8-bit ports (Address, Write_Data, Read_Data) are used in order to have access to the configuration register of a PnP plug-in card; see also Figure 1. These ports are shown in Table 1. Commands can be implemented from a program which requests the status, configures the Plug-and-Play hardware, and which has access to the data stored in the EEPROM.

PORT NAME	ADDRESS	TYPE
ADDRESS	0x279 (Printer Status Port)	Write only
WRITE_DATA	0xA79 (Printer Status Port + 0x800)	Read only
READ_DATA	Can be shifted to area 0x203 to 0x3FF	Read only

Table 1: Autoconfiguration Ports

Every card must provide information which describes the individual characteristics of the PnP plug-in card in question. This information is stored in an EEPROM and, as already mentioned, is read out from the Plug-and-Play program. After that, the configuration of the hardware for the individual card can be determined. As shown in Figure 1, for this purpose the registers “Card Control”, “Logical Device Control” and “Logical Device Configuration” are available. On every PnP plug-in card there is a precise Card Control register, which is used for the overall data of the complete card. However, each logic unit also possesses both a Logical Device Control register and a Logical Device Configuration register. The Logical Device Control register controls the logic unit, such as switching the logic unit onto the ISA bus. The Logical Device Configuration register contains the information about the resources of the logic unit. This means that a dual register is needed for the TL16PNP100A, and a single register for the TL16PNP550A.

In the following Section, further details will be given of the design of a Windows® 95 compatible Plug-and-Play COM port, using the TL16PNP550A.

3. TL16PNP550A

The TL16PNP550A is a combination component, which integrates a Plug-and-Play controller and a serial interface, compatible to the serial interface (UART) TL16C550C. Figure 2 shows the block circuit diagram of this component. It can be seen that there are two separate blocks for the Plug-and-Play controller and the serial interface (UART). This device can be used in various operating configurations. The Plug-and-Play controller can be utilized together with the integrated interface (UART) to provide a Plug-and-Play compatible COM port. The two functional blocks - serial interface (UART) and PnP controller - can also be used separately. For example, the PnP controller can control an external logic unit, and the serial interface (UART) is used as a standard interface.

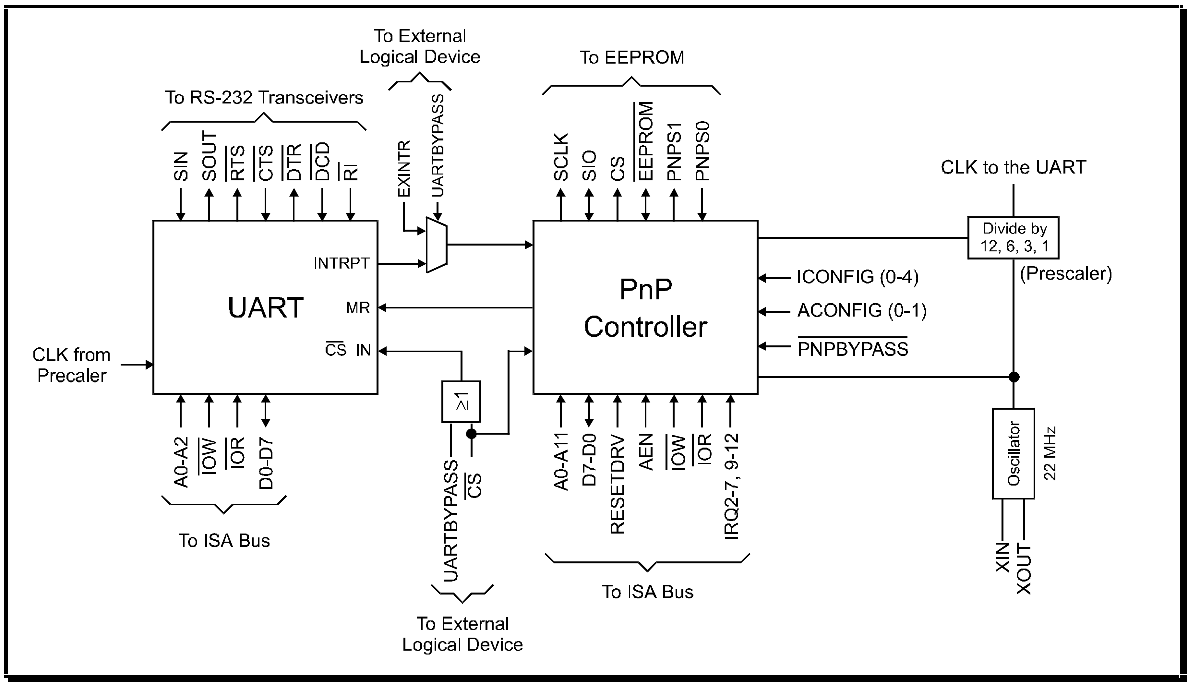

Figure 2: Block circuit diagram of the TL16PNP550A

A Plug-and-Play-COM port which is 16550 Windows® 95 compatible and uses this component will now be described.

4. 16550 compatible COM port

Figure 3 shows the circuit of the 16550 Windows® 95 compatible interface. The hardware requirements for an interface of this kind are very modest. The integrated circuit TL16PNP550A is used, and this needs an external system clock. In this circuit an external EEPROM is also needed, in order to store the configuration data of the Plug-and-Play compatible ISA card.

A PC card which is designated 'Windows® 95 compatible' and carries the WIN95 logo must be able to decode a 16-bit I/O address. Since the TL16PNP550A can only decode the 10 lower address bits, an OR gate is needed to decode the addresses A10 - A15 and the signal AEN. The output of the OR gate is then at a Low level when the addresses A10 - A15 and AEN are also at a Low level.

The outputs D0 - D7, IRQ3 - IRQ7, IRQ9 - IRQ12 and IRQ15 provide a current I_{OL} = 24 mA at $V_{OL} \leq 0.4$ V, and I_{OH} = -12 mA at $V_{OH} \geq 2.4$ V. The connection to the ISA bus can therefore be made without the use of an additional buffer. The level shifting at the serial interface is performed by means of V.28 compatible transmitter and receiver circuits.

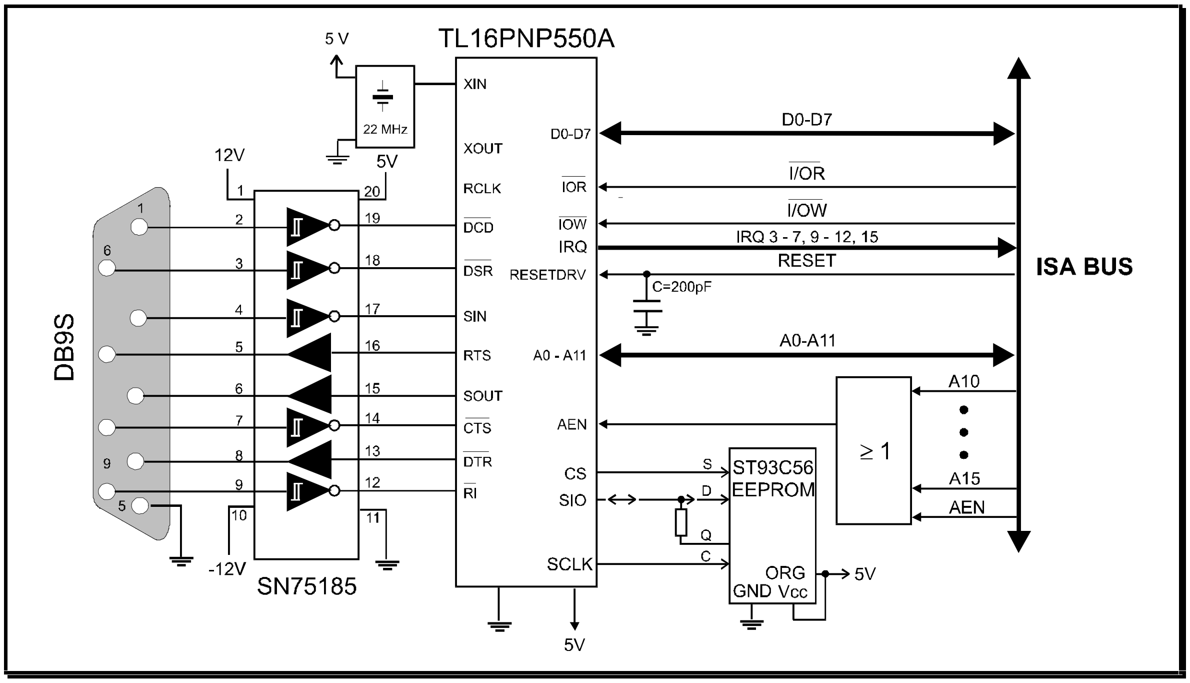

Figure 3: Circuit of the 16550 Windows® 95 compatible COM port.

The design of this interface will usually present no particular problems. The question will, however, arise as to what should be loaded into the EEPROM, so that the corresponding program - such as Windows® 95 - can recognize and configure the PC card correctly. This subject will be discussed in the next section.

5. EEPROM Programming

As shown in Figure 1, the TL16PNP550A provides an interface to the EEPROM ST93C56/66. The EEPROM should contain the configuration data of the Plug-and-Play ISA card. The following listing shows a programming example for a 16550 Windows® 95 compatible COM port.

Address	**Value**	**Description**
		Clock Prescaler:
0x00	0xFF	
0x01	0x3F	Divisor Value, bits[15:14]
		Serial Identifier:
0x02	0x43	Vendor ID byte 0
0x03	0x11	Vendor ID byte 1
0x04	0x43	Vendor ID byte 2
0x05	0x04	Vendor ID byte 3
0x06	0x00	Serial number byte 0
0x07	0x01	Serial number byte 1
0x08	0x00	Serial number byte 2
0x09	0x02	Serial number byte 3
0x0A	0x1E	1. check sum of vendor ID and serial number (byte 0x02 to byte 0x09)
		Plug-and-Play Version Number:
0x0B	0x0A	Plug-and-Play version number descriptor
0x0C	0x10	Version in packed BCD format, example: Version 1.0
0x0D	0x01	Vendor specific version number
		Identifier String:
0x0E	0x82	Identifier string descriptor
0x0F	0x0F	Length byte 0 (here: 15 decimal)
0x10	0x00	Length byte 1
0x11	"16550A COM PORT"	Identifier String
		Logical Device ID:
0x20	0x15	Logical device ID descriptor

0x21	0x43	Logical device ID byte 0
0x22	0x11	Logical device ID byte 1
0x23	0x43	Logical device ID byte 2
0x24	0x04	Logical device ID byte 3
0x25	0x02	Logical device flags
		Compatible Device ID:
0x26	0x1C	Compatible device ID descriptor
0x27	0x41	Compatible device ID byte 0
0x28	0xD0	Compatible device ID byte 1
0x29	0x05	Compatible device ID byte 2
0x2A	0x01	Compatible device ID byte 3
		Interrupt:
0x2B	0x22	Interrupt descriptor
0x2C	0x78	IRQ mask bits[7:0], bit[0] = IRQ0 ...
0x2D	0x9e	IRQ mask bits[15:8], bit[8] = IRQ8 ...
		I/O Ports:
0x2E	0x47	I/O port descriptor
0x2F	0x01	Information, bit[0] = 1, 16 bit decoding
0x30	0x00	Range minimum base address, bits[7:0]
0x31	0x02	Range minimum base address, bits[15:8]
0x32	0xF8	Range maximum base address, bits[7:0]
0x33	0x03	Range maximum base address, bits[15:8]
0x34	0x08	Alignment for minimum base I/O address that the card may be configured for
0x35	0x08	The number of contiguous I/O ports requested
		End Tag:
0x36	0x79	End Tag descriptor
0x37	0xC3	2. check sum (If the check sum field is zero, the resource data is treated as if it checksummed properly. Configuration proceeds normally)

The configuration data will be described in detail in the following Section.

5.1 Clock Prescaler

The oscillator frequency provided to the circuit TL16PNP550A should be between 10 and 22.1184 MHz. So that the next circuit receives the desired clock frequency, the frequency of the oscillator must be divided appropriately; the standard frequency of a COM port is 1.8432 MHz. The TL16PNP550A contains a functional block which can perform this division. The desired division factor is stored in the EEPROM at Address 0. The two highest value bits of this address define the division factor, as shown in Table 2.

EEPROM address 000 (Bits 15 and 14)	Division Factor
00	12
01	6
10	3
11	1

Table 2: Clock Division Factor

A 22.1184-MHz quartz crystal was used in the application described. This results in a division factor of 12 ($12 \times 1.8432 = 22.1184$ MHz), and the highest value bits in the EEPROM Address 000 must be set to 0.

5.2 Serial Identifier

When the computer is switched on, all Plug-and-Play cards have the same I/O port address. For this reason it is necessary to be able to isolate each card individually; this is only possible with a unique identification process, performed with the help of serial identifier. The access to this identification is made serially, and bit-by-bit. The identification consists of a combination which is 72 bit in length, and is made up of the 4 byte long manufacturer recognition, the 4 byte long serial number, and the 8-bit long check sum. The first two bytes of the manufacturer recognition are allocated centrally, in order to ensure that there is a unique identification for every card which has been developed.

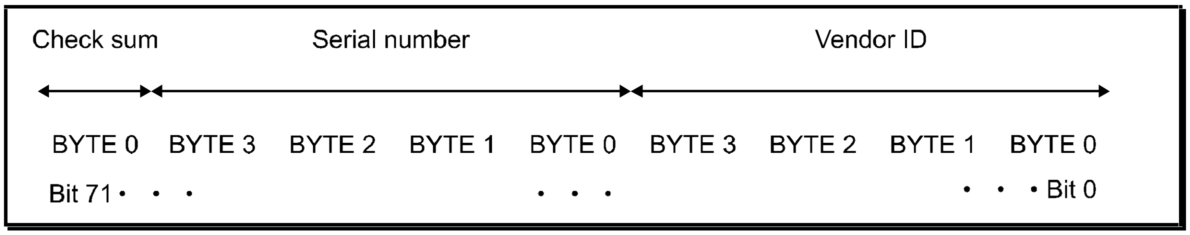

Figure 4: Constitution of the serial identifier

The check sum is derived from the vendor ID (byte 0 to byte 3) and the serial number (byte 0 to byte 3) of the serial identifier. This check sum is used to ensure that no conflict has arisen during read-out of the identification. An appropriate algorithm is used in order to compare the check sum. This algorithm is provided by logic (LFSR, Linear Feedback Shift Register) contained in the Plug-and-Play controller: see also Figure 1. Figure 5 shows the construction of the Linear Feedback Shift Register.

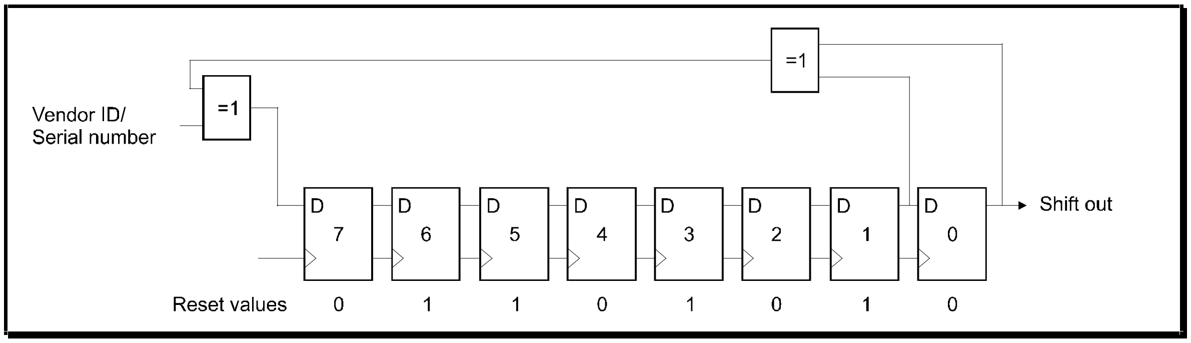

Figure 5: Construction of the LFS Register

The LFS register consists of an 8-bit long shift register. The signal LFSR[0] ⊕ LFSR[1] ⊕ manufacturer recognition / serial number is led back to LFSR[7]. This register is initially loaded with the value $6A_{16}$. After 64 read accesses / shift operations, the LFS register is loaded with the correct check sum, and this is now compared with the check sum stored in the EEPROM. The manufacturer recognition and serial number used in the application, chosen at random in this example, result in a check sum of $1E_{16}$.

5.3 Plug-and-Play Version Number

The Plug-and-Play version number gives the version of the Plug-and-Play standard with which the PnP plug-in card is compatible. In addition, it provides a version number specific to the manufacturer, which can be used by the equipment driver to recognize the card version.

Byte 0	Value = 00001010b (Type=0, abbreviated recognition of the Plug-and-Play Version (0x1) and length of the information in bytes (2 byte)
Byte 1	Plug-and-Play Versions Number (BCD format, bits[7:4].bits[3:0]; Example: Version 1.0 = 0x10, Version 3.5 = 0x35
Byte 2	Version number specific to the manufacturer

Table 3: Coding of the Plug-and-Play version number

Table 3 gives details of coding information. Byte 0, bit[7] gives the length of the information, whereby Type = 0 indicates a length of 2 to 8 byte, and Type = 1 a length of up to 64 Kbyte. The abbreviated recognition indicates that the subsequent information identifies the Plug-and-Play version.

5.4 Identifier String

It is possible to store a character string in the EEPROM which will appear on the screen of the computer when the PnP plug-in card is automatically configured. Table 4 describes the format of this identification text - the "Identifier String".

Byte 0	Value = 10000010b (Type=1, abbreviated recognition of the identification text
Byte 1	Bits[7:0] give the lower eight bits of the string length
Byte 2	Bits[15:8] give the upper eight bits of the string length
n Bytes	String in ANSI format, which describes the PC card

Table 4: Format of the identifier string

In the application presented, the character string “16550A COM Port” was chosen to describe the PnP plug-in card in more detail.

5.5 Logical Device ID

The format for identifying the logical device (Logical Device ID) is identical with that used for serial identification (Serial Identifier). The identification serves to make an appropriate choice of driver. In contrast to serial identification (Serial Identifier), the identification of the logical device (Logical Device ID) does not need to be unique: this means that the same identification may appear in connection with another logical device. A PnP plug-in card with two COM ports may therefore have the same identification for each COM port.

Byte 0	Value = 00010101b (Type=0, abbreviated recognition for the identification of the logical device, length = 5
Byte 1	Logical device ID bits[7:0]
Byte 2	Logical device ID bits[15:8]
Byte 3	Logical device ID bits[23:16]
Byte 4	Logical device ID bits[31:24]
Byte 5	Bits[7:1], if set, show which configuration registers from 0x31 to 0x37 are supported by the logical device.

	Bit[0], if set, shows that the logical device is able to participate in the booting process.
Byte 6	Bits[7:0], if set, show which configuration registers from 0x38 to 0x3F are supported by the logical device.

Table 5: Identification of the logical device

Table 5 gives individual information. In the application presented using the TL16PNP550A, byte 5 must be set to 0x02 for the identification of the logical device (Logical Device ID), because this component only supports the register 0x37 (I/O Range Check). For this reason, byte 6 is also not used.

5.6 Compatible Device ID

With the identification of a compatible logical device (Compatible Device ID), it is possible to give the identification of other logical devices with which the plug-in card is compatible.

Byte 0	Value = 00011100b (Type=0, abbreviated recognition of the identification of the compatible logical device, length = 4
Byte 1	Compatible device ID bits[7:0]
Byte 2	Compatible device ID bits[15:8]
Byte 3	Compatible device ID bits[23:16]
Byte 4	Compatible device ID bits[31:24]

Table 6: Format of the identification of the compatible logical device

Table 6 shows the format of the identification of a compatible logical device (Compatible Device ID). To give an example, a card manufacturer brings out a PnP plug-in card with the Logical Device ID 0xABCD0000. At a later date, a newer PnP plug-in card is announced with the Logical Device ID 0xABCD0001, which however is 100% compatible to the older card. For this new logical device, the Compatible Device ID 0xABCD0000 can be allocated. In this way the equipment driver ID 0xABCD0001 will be loaded, if it can be found. If not, then the driver for the ID 0xABCD0000 is loaded for the logical device.

A list is available (see References in Section 11) which gives the identification of the better known PC cards. The application described is concerned with a Plug-and-Play 16550 compatible COM port. In the list mentioned, an identification ‘PNP0510’ is defined for this card. The corresponding Compatible Device ID is then known as ‘41D00501’.

5.7 Interrupts

The interrupt masking indicates which interrupts must be taken into account during allocation. It should be possible for the corresponding interrupts to be supported by the Plug-and-Play controller. The allocation of the possible 16 interrupts is undertaken by two masks with a length of 8 bit.

Byte 0	Value = 0010001xb (Type=0, abbreviated recognition of the IRQ data, length = 2 or 3
Byte 1	IRQ mask, bits[7:0]. Bit[0] = IRQ0, bit[1] = IRQ1, …
Byte 2	IRQ mask, bits[15:8]. Bit[0] = IRQ8, bit[1] = IRQ9, …
Byte 3	IRQ information. If set, each bit indicates the ability to support a specific type of interrupt. This information contained in byte 3 is optional. If it is not available, ISA compatibility is assumed (edge - triggering, interrupt active High). Bit[7:4] are reserved, and must be set to 0 Bit[3] level triggering, active Low Bit[2] level triggering, active High Bit[1] edge triggering, active Low Bit[0] edge triggering, active High (ISA compatible)

Table 7: Format of the Interrupt Data

Table 7 shows the format of the Interrupt data. With the circuit TL16PNP550A it is possible to select Interrupts IRQ3 - IRQ7, IRQ9 - IRQ12 and IRQ15. In the application described, byte 3 of the Interrupt data was not needed, since an ISA standard compatible PC card was involved.

5.8 I/O Ports

On starting the computer, the I/O ports needed by a Plug-and-Play compatible PC card are allocated. A procedure is followed in which a check is made as to whether the I/O port which has been allocated can lead to conflicts with another PC card. For this reason the possibility is provided to allocated an address range to the Plug-and-Play compatible plug-in card, so that should an address conflict arise, a new address can be sought. This address range is defined in the I/O port description.

Byte 0	Value = 01000111b (Type=0, abbreviated recognition of the I/O port description, length = 7
Byte 1	Bits[7:1] are reserved and must be set to 0 Bit[0] indicates, if it has been set, that the logical device can decode the full 16 bit ISA address. If bit[0] is not set, this indicates that the logical device is only decoding the address bits[9:0].
Byte 2	Bits[7:0] of the minimum base I/O address, in which the PC card can be configured.
Byte 3	Bits[15:8] of the minimum base I/O address, in which the PC card can be configured.
Byte 4	Bits[7:0] of the maximum base I/O address, in which the PC card can be configured.
Byte 5	Bits[15:8] of the maximum base I/O address, in which the PC card can be configured.
Byte 6	Byte 6 defines the alignment for minimum base address, increment in 1 byte blocks.
Byte 7	The information contained in byte 7 gives the number of the successive I/O ports requested.

Table 8: Format of the I/O port information

Table 8 shows the format of the I/O port information. The 16550A compatible COM port presented in this Applications Report can be configured in the address range from 0x200 to 0x3FF. This information is stored in bytes 2 - 5.

It has already been mentioned in Section 4 that the TL16PNP550A includes 10-bit address decoding. However, for a WIN95 Logo, Windows® 95 requires 16-bit address decoding. For this reason and as discussed above, an OR gate has been integrated into the circuit in question, in order to be able also to decode the higher addresses. Therefore, bit[0] must be set in byte 1 of the I/O port information. This indicates that the PnP plug-in card is able to decode a 16-bit address.

The serial interface in the TL16PNP550A occupies eight successive I/O ports. byte 6 and byte 7 of the I/O port information are thus set to 8.

5.9 End Marking

The end marking (End Tag) indicates the end of the resource data.

Byte 0	Value = 01000111b (Type=0, abbreviated recognition of the end marking.
Byte 1	2. check sum, which is generated from the data after the serial identification.

Table 9: Format of the end marking (End Tag)

Table 9 shows the format of the end marking (End Tag). Byte 0 contains the abbreviated recognition for this marking, and byte 2 contains a 2nd. check sum. This check sum is formed by first calculating the sum from the contents of the EEPROM beginning with memory location 0xA to 0x36 (end marking); in this example, the result is a sum of $93D_{16}$. The lower eight bits (3D) of this sum are subtracted from the number 100_{16} (complement to two). The result of this provides the check sum, which in this example is $C3_{16}$.

In the next Section, further details will be given of the construction of the TL16PNP100A.

6. TL16PNP100A

The TL16PNP100A is a Plug-and-Play controller with which two logical devices can be controlled. The block circuit diagram of this Plug-and-Play controller is shown in Figure 1 of this Applications Report. A typical application can be seen in Figure 6. The TL16PNP100A provides two chip-select signals $\overline{CS0}$ and $\overline{CS1}$ for two logical devices, which are then switched on to the ISA bus.

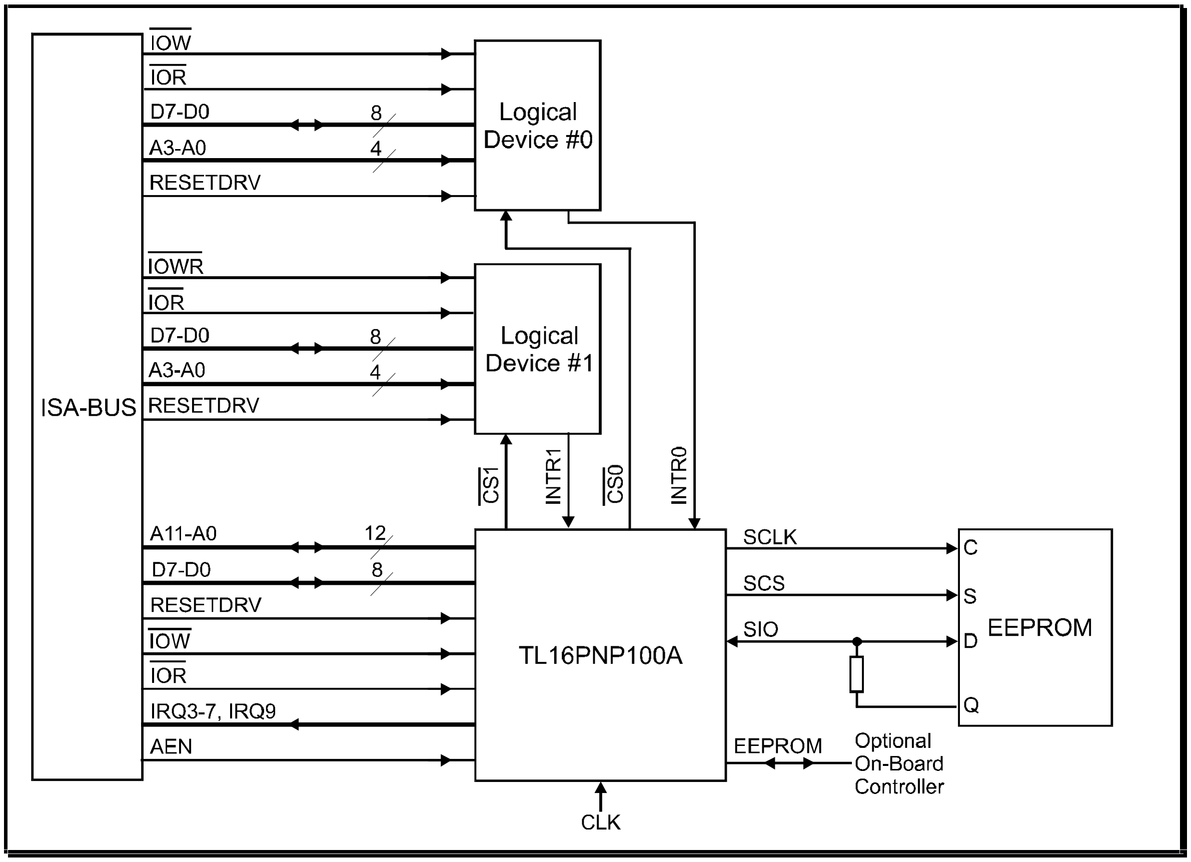

Figure 6: Typical application of the TL16PNP100A

The interrupt requirements of the logical devices are passed on to the Plug-and-Play controller, which in turn switches these signals on to the ISA bus. An external EEPROM is also necessary with the TL16PNP100A, in order to store the configuration data of the Plug-and-Play compatible ISA card, together with the block sizes of the logical devices.

The next part of this Applications Report describes how the TL16PNP100A can be used to construct a double 16550 Windows® 95 compatible Plug-and-Play COM port.

7. Circuit diagram of the dual-channel COM port

The circuit in Figure 7 shows a dual-channel 16550 Windows® 95 compatible COM port using the TL16PNP100A as a Plug-and-Play controller.

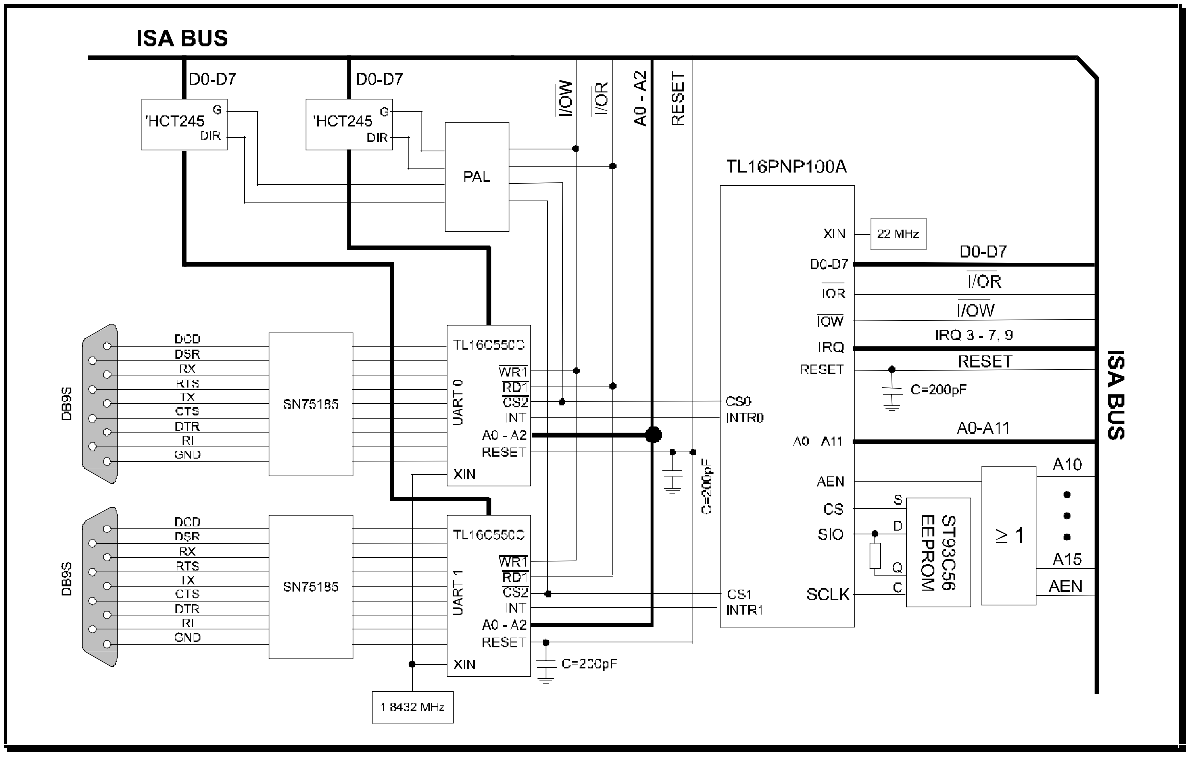

Figure 7: Circuit diagram of the dual-channel COM port

The hardware requirements for this interface are fairly complex, since the TL16PNP100A does not possess an integrated serial interface (UART). The connection of the Plug-and-Play controller to the ISA bus is however identical to that in the circuit with the TL16PNP550A. Besides the clock generator, an EEPROM is needed in this application, and stores the configuration data of the two serial interfaces.

As already mentioned in Section 4, a PC card which is designated as “Windows® 95 compatible” and is intended to display the WIN95 logo, must be able to decode a 16-bit I/O address. As with the TL16PNP550A, the TL16PNP100A can only decode the lower 10 bits of the ISA address bus. For this reason and as described in the previous application, an OR gate is needed to decode the addresses A10 - A15, and also the signal AEN. The output of the OR gate then switches to a Low level only if the addresses A10 - A15 and AEN are also at a Low level.

The two Chip-Select outputs $\overline{\text{CS0}}$ and $\overline{\text{CS1}}$ of the TL16PNP100A control the Chip-Select inputs of the two circuits TL16C550C (UARTs). These are each connected via a bi-directional line driver SN74HCT245 to the ISA bus. The control of the line driver is performed by a PAL.

The expression for this is as follows:

$$\overline{\text{G}} = (\overline{\text{CS}} \vee \overline{\text{I / OW}} \wedge \overline{\text{I / OR}})$$

$$\text{DIR} = \overline{(\overline{\text{CS}} \vee \overline{\text{I / OR}})}$$

The level conversion to the outside world is achieved by means of a V.28 compatible transmitter/receiver. However, problems with the programming of the EEPROM arise more frequently than in the application previously described. The next Section therefore goes into more detail concerning the programming of the PnP plug-in card in conjunction with the TL16PNP100A.

8. EEPROM Programming

The Plug-and-Play controller TL16PNP100A provides an interface to the EEPROM ST93C56/66. The EEPROM contains the block sizes of the logical devices, and the configuration data of the Plug-and-Play ISA card. The following listing shows a programming example for a dual channel 16550A Windows® 95 compatible COM port.

Address	**Value**	**Description**
		Block Size:
0x00	0x00	
0x01	0x88	Programmable block Size of the logical device
		Serial Identifier:
0x02	0x43	Vendor ID byte 0
0x03	0x11	Vendor ID byte 1
0x04	0x43	Vendor ID byte 2
0x05	0x04	Vendor ID byte 3
0x06	0x00	Serial number byte 0
0x07	0x01	Serial number byte 1
0x08	0x00	Serial number byte 2
0x09	0x02	Serial number byte 3
0x0A	0x1E	1. check sum of vendor ID and serial number (byte 0x02 to byte 0x09)
		Plug-and-Play Version number:
0x0B	0x0A	Plug-and-Play version number descriptor Versionsnummer
0x0C	0x10	Version in packed BCD format, example: Version 1.0
0x0D	0x01	Vendor specific version number
		Identifier String:
0x0E	0x82	Identifier string descriptor
0x0F	0x13	Length byte 0 (here: 19 decimal)
0x10	0x00	Length byte 1
0x11	"2 x 16550A COM PORT"	Identifier String

		Logical Device ID - 1st logical device:
0x24	0x15	Logical device ID descriptor
0x25	0x04	Logical device ID byte 0
0x26	0x43	Logical device ID byte 1
0x27	0x00	Logical device ID byte 2
0x28	0x01	Logical device ID byte 3
0x29	0x02	Logical device flags
		Compatible Device ID - 1st logical device:
0x2A	0x1C	Compatible device ID descriptor
0x2B	0x41	Compatible device ID byte 0
0x2C	0xD0	Compatible device ID byte 1
0x2D	0x05	Compatible device ID byte 2
0x2E	0x01	Compatible device ID byte 3
		Interrupt - 1st logical device:
0x2F	0x22	Interrupt descriptor
0x30	0xf8	IRQ mask bits[7:0], bit[0] = IRQ0 ...
0x31	0x02	IRQ mask bits[15:8], bit[8] = IRQ8 ...
		I/O Ports 1st logical device:
0x32	0x47	I/O port descriptor
0x33	0x01	Information, bit[0] = 1, 16 bit decoding
0x34	0x00	Range minimum base address, bits[7:0]
0x35	0x02	Range minimum base address, bits[15:8]
0x36	0xF8	Range maximum base address, bits[7:0}
0x37	0x03	Range maximum base address, bits[15:8]
0x38	0x08	Alignment for minimum base I/O address that the card may be configured for
0x39	0x08	The number of contiguous I/O ports requested
		Logical Device ID - 2nd logical device:
0x3A	0x15	Logical Device ID descriptor
0x3B	0x04	Logical Device ID byte 0
0x3C	0x43	Logical Device ID byte 1
0x3D	0x01	Logical Device ID byte 2

0x3E	0x01	Logical Device ID byte 3
0x3F	0x02	Logical device flags
		Compatible Device ID - 2nd logical device:
0x40	0x1C	Compatible device ID descriptor
0x41	0x41	Compatible device ID byte 0
0x42	0xD0	Compatible device ID byte 1
0x43	0x05	Compatible device ID byte 2
0x44	0x01	Compatible device ID byte 3
		Interrupt - 2nd logical device:
0x45	0x22	Interrupt descriptor
0x46	0xf8	IRQ mask bits[7:0], bit[0] = IRQ0 ...
0x47	0x02	IRQ mask bits[15:8], bit[8] = IRQ8 ...
		I/O Ports 2nd logical device:
0x48	0x47	I/O port descriptor
0x49	0x01	Information, bit[0] = 1, 16 bit decoding
0x4A	0x00	Range minimum base address, bits[7:0]
0x4B	0x02	Range minimum base address, bits[15:8]
0x4C	0xF8	Range maximum base address, bits[7:0}
0x4D	0x03	Range maximum base address, bits[15:8}
0x4E	0x08	Alignment for minimum base I/O address that the card may be configured for
0x4F	0x08	The number of contiguous I/O ports requested
		End Tag:
0x50	0x79	End Tag descriptor
0x51	0x32	2. check sum (If the check sum field is zero, the resource data is treated as if it checksummed properly. Configuration proceeds normally)

There is no difference in the configuration data when compared with the previous application with the TL16PNP550A. The only difference is that, when using the TL16PNP100A, no division factor needs to be stored in the EEPROM. Instead of this, a block size for the logical devices must be stored in the EEPROM.

8.1 Block Size

Besides storing the resources of the card, when using the TL16PNP100A the block size of the individual logical devices must also be stored in the EEPROM. After switching on the computer, this block size will be read out, beginning at address 0 in the EEPROM. Table 10 shows the possible block sizes together with the corresponding coding.

DATA [15:13]/[11:9]	**Block Size**	**Coded Address Bits**
000	1 byte	[A9:A0]
001	2 byte	[A9:A1]
010	4 byte	[A9:A2]
100	8 byte	[A9:A3]
111	16 byte (default)	[A9:A4]

Table 10: Block Sizes

The data bits [15:13] contain the block size of the first logical device, whereas the data [11:9] contain the block size of the second logical device.

9. References

Texas Instruments, Data Sheet TL16PNP550A SLLS190A

Texas Instruments, Data Sheet TL16PNP100A SLLS200A

Texas Instruments, Data Book Data Transmission Circuits SLLD001A

Texas Instruments, Data Book Data Transmission Circuits SLLD003

Texas Instruments, Data Sheet SN74HCT245 SCLS020B

Texas Instruments, Data Sheet TIBPAL16L8 SRPS006D

Intel Corporation / Microsoft Corporation, Plug-and-Play ISA Specification, Version 1.0a, May 5, 1994

Microsoft, PC card identification,
http://www.microsoft.com/hwdev/download/devids.txt