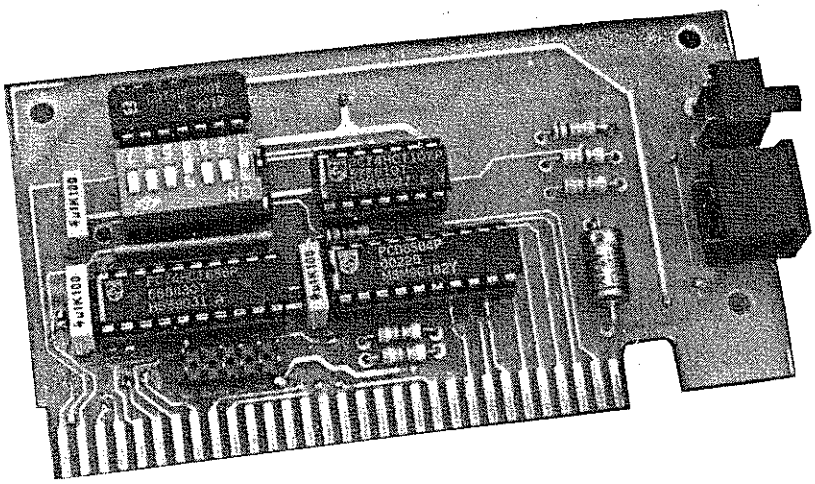
I²C INTERFACE FOR PCs

FRONT COVER PROJECT



The I²C interface (Inter-IC Communication) is a Philips invention that has been in use for years to enable ICs to communicate with each other in complex electronic equipment such as radios, video recorders and television sets. The insertion card we present here puts your PC in control of about ten I²C compatible ICs.

by J. Ruffel

IT is a simple but well known fact that the requirement for external pin connections is a limiting factor as regards the complexity of integrated circuits. Also, the more pins on an IC, the more expensive the device becomes to package and mount on a printed circuit board. Further, a large number of connections is inevitably coupled to a higher risk of malfunctions. No wonder IC manufacturers have sought alternative ways to allow complex ICs to communicate with as few as possible interconnections.

The I²C bus designed by Philips is such an alternative: it allows ICs to exchange data via two wires. This type of (serial) communication is particularly suited to relatively slow data transfer, and the protocol certainly does not allow, say, a computer RAM card to be implemented with I²C devices. By contrast, the I²C bus and protocol are perfect for, say, an I/O port or a real-time clock in a video recorder.

The I²C card discussed here allows an MS-DOS compatible PC to communicate with I²C ICs in external application circuits. As such, the card is for all of you who have noticed the large potential of I²C ICs (they are not generally expensive because of mass production), but have so far lacked the means to set up a control link to them. Having built the I²C interface, programming languages such as C, Pascal or assembler may be used to communicate with the I²C ICs as if they were 'external devices'.

I²C, a powerful standard

The I²C bus is a system bus based on three signals: SDA (system data), SCL (system clock) and ground. The SDA and SCL lines are of the open-drain type, and must be tied to the positive supply line via an external resistor to create a bus structure that allows multiple inputs and outputs to be connected in parallel.

Figure 1 shows the basic electrical configuration. The two communication lines are logic high when they are inactive. The number of ICs connected to the bus is, in principle, unlimited. Note, however, that the lines do have a maximum specification of 400 pF in respect of the load capacitance. The maximum data rate that can be achieved on the I²C bus is about 100 kBit/s.

The definitions used in relation to I²C bus functions are basically as follows:

Master: this is the IC that determines the timing and the direction of a data transfer. This IC is the only one on the I²C bus to supply clock pulses on the SCL line. When multiple master devices are connected to a single I²C bus, this set-up is called a multi-master system.

Slave: this is any IC connected to the I²C bus that is not capable of generating clock pulses. Slave ICs receive commands and clock signals from a master.

Bus free: the bus is free when SDA and SCL are both logic high. A master can access the bus only when this is free.

Start: a master occupies the bus by generating a start condition, which means that SDA is made low while SCL remains high.

Stop: a master can free the bus again by generating a stop condition, which means that SDA is made high while SCL is high.

Data valid: the data on the SDA line must be

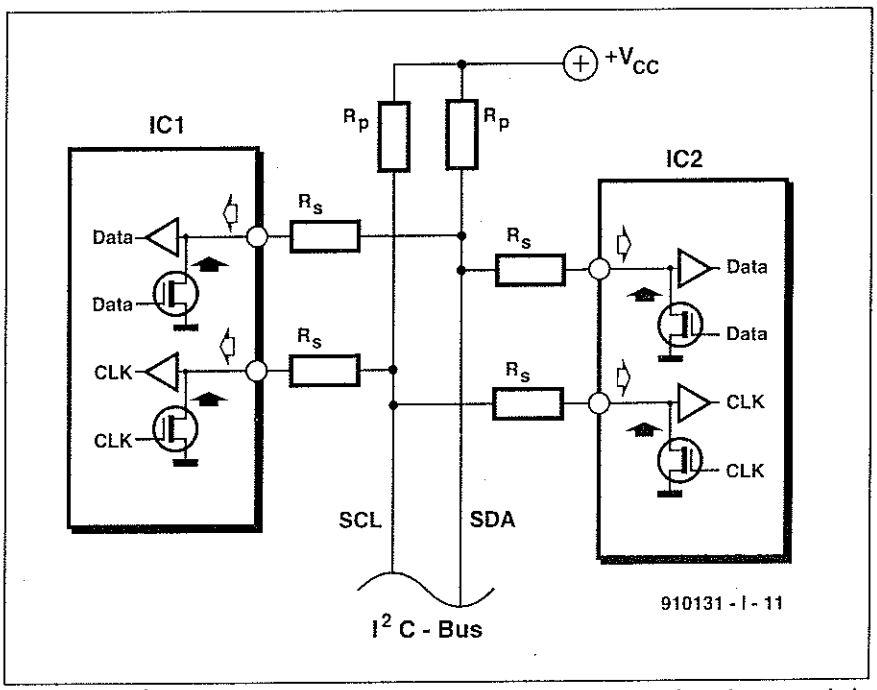


Fig. 1. The I²C bus is basically a 2-wire serial communication system based on open-drain outputs in connected devices, series resistors (R_S) and pull-up resistors (R_P).

stable while SCL is high. This renders the start and stop conditions unique.

Data format: each 'chunk' of information sent via the I²C bus consists of eight data bits (one byte). Each byte is followed by a ninth clock pulse, during which the receiving IC (a master or a slave) is to generate an acknowledge (ACK) pulse. This is done by making the SDA line low during the ninth clock pulse.

IC address: each IC that is suitable for use on the I²C bus has its own, unique, address, which is determined by the manufacturer. In general, this address is 'burned' into the IC, although there are also ICs that allow a part of the address to be set externally. This option allows a number of ICs of the same type to be connected to a single bus. Address 00 is the 'general call address', to which all ICs on the bus respond.

R/$\overline{W}$ bit: every IC has a 7-bit address. The eighth bit (LSB) that is sent during the addressing operation, is the R/$\overline{W}$ (read/write) bit. If this bit is '1', a master device reads data from a slave device. When it is '0', a master device writes data to a slave device.

Bus protocol

A protocol has been drawn up to initiate the communication between two ICs on the I²C bus. This protocol is briefly described below.

As soon as the bus is free, a master can occupy it by generating a start condition. The first byte transmitted after the start condition contains the 7-bit IC address and the R/$\overline{W}$ bit. If the addressed IC is present (i.e., connected to the bus), it responds by returning an ACK pulse. After that, the data exchange may commence.

When the R/$\overline{W}$ pulse was '0' previously, the master sends data to the slave until it no longer receives ACK pulses, or until all data has been transmitted.

When the R/$\overline{W}$ pulse was '1' previously, the master generates clock pulses, during which the slave is allowed to send data. After every received byte, the master (which is a receiving device at this stage) generates an ACK pulse. This continues until the master no longer supplies ACK pulses.

The master can free the bus again by generating a stop condition. If it wants to continue communicating, it is possible that the master generates a new start condition instead of a stop condition. This new start condition is called a repeated start, and can be used to address a different IC, or make the R/$\overline{W}$ bit toggle. Figure 2 shows the start and stop conditions in a timing diagram.

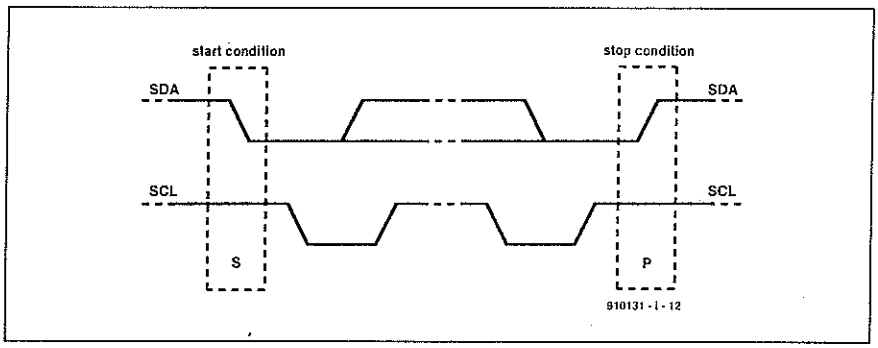


Fig. 2. Signal levels on the SDA and SCL lines mark start and stop conditions.

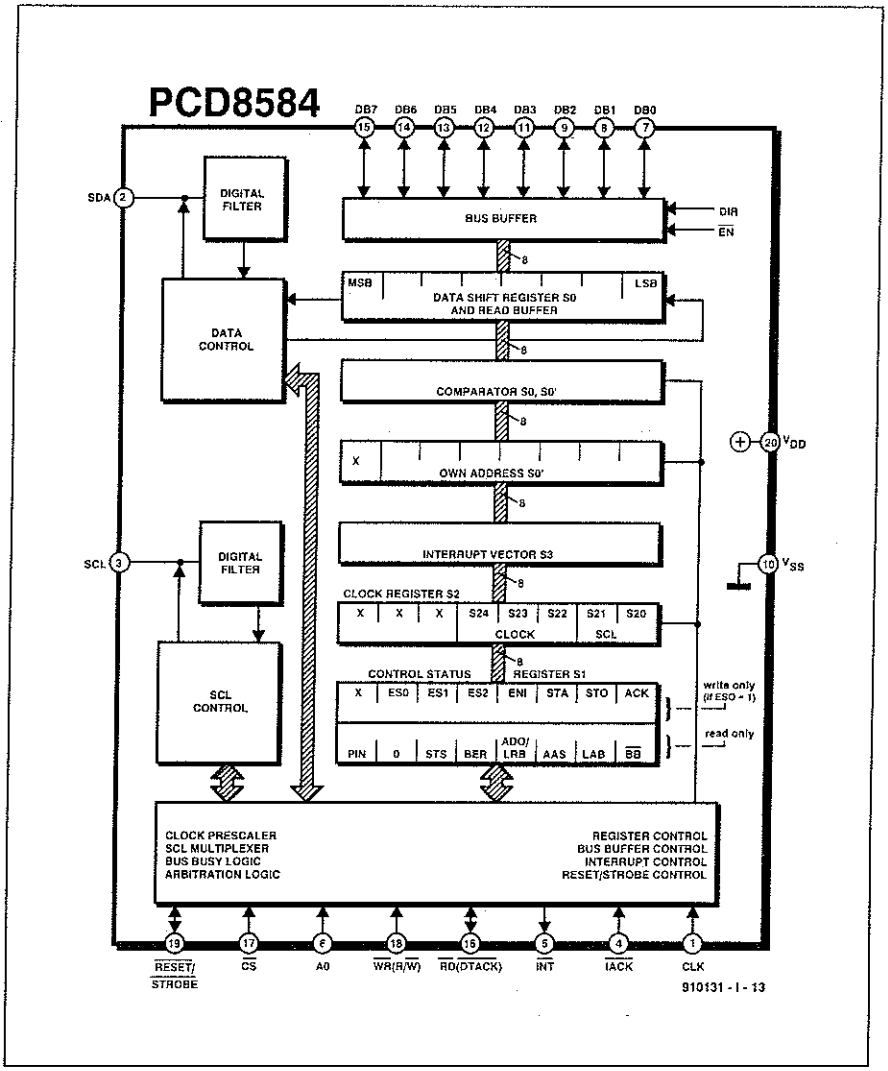


Fig. 3. The PCD8584 I²C controller is a quite complex IC — here, the block diagram is shown.

Philips Components and a number of other manufacturers have a wide range of I²C compatible ICs available, including RAMs, EEPROMs, microcontrollers, I/O ports, DTMF encoders, infra-red transmitters and receivers, ADC and DACs, and a real-time clock with calendar (for an overview, see Ref. 1). The range of I²C devices is continuously expanded with new ICs.

The PCD8584 is an I²C bus controller specifically developed to simplify the communication with a parallel port. This IC, which is at the heart of the present interface, arranges all control actions required on the bus, and thus allows ready communication between an I/O port of a PC, and I²C devices.

The PCD8584 I²C controller

The PCD8584 is a powerful, universal, I²C bus controller that forms the link between an 8-bit parallel port of a microcontroller or microprocessor, and the serial I²C bus. The IC supports reading and writing of bytes via the I²C bus, and is remarkably simple to implement in systems based on a different processor types, such as the 8048/8051 controllers, 80xx processors, and the 68000.

The block diagram of the PCD8584, Fig. 3, shows a number of functional blocks:

Bus buffer. This is the circuit between the computer bus and the shift register in the PCD8584.

S0', own address. In a multi-master system, this register contains the address to which the controller responds. This address may not be '00' unless the controller is to work in

PCD8584 I²C CONTROLLER

The information below is supplementary to that contained in the PCD8584 datasheet, and aimed at those of you who intend to work on the device driver.

Initialization

After the circuit has been reset, the controller must be set to the correct mode (80xx or 68000). The default mode is 80xx compatible. On detecting a high-to-low transition at the $\overline{WR}$ input, while $\overline{CS}$ is high, the PCD8584 switches to 68000 mode. Next, a value other than 0 must be put into register S0'. Programming a 0 results in the IC switching to 'passive monitor mode' in which data on the I²C bus is indiscriminately put into the read buffer.

After the mode selection, the internal prescaler is programmed via register S2. This is done with the aid of bits S20 to S24. Table 1 shows the function of the bits in relation to the available clock frequencies.

The I²C interface card uses two addresses in the I/O range of the PC. All registers contained in the controller can be accessed via these two addresses. First, bits ES0, ES1 and ES2 in the 'S1: control' register must be programmed to give the desired settings of the I²C bus controller. The most useful settings are indicated with asterisks in Table 2.

Table 1. PCD8584 SCL and clock frequency selection

SCL frequency			Internal clock frequency			
S21	S20	f_{SCL} (kHz)	S24	S23	S22	f_{CLK} (MHz)
0	0	90	0	x	x	3
0	1	45	1	0	0	4.43
1	0	11	1	0	1	6
1	1	1.5	1	1	0	8
			1	1	1	12

Table 2. Control/Status register S1

ES0 = 0: serial interface OFF

A0	ES1	ES2	$\overline{IACK}$	Function	
1	x	x	x	R/W S1: CONTROL	*
0	0	0	x	R/W S0': (own address)	*
0	0	1	x	R/W S3 (interrupt vector)	
0	1	0	x	R/W S2 (clock register)	*

ES0 = 1: serial interface ON

A0	ES1	ES2	$\overline{IACK}$	Function	
1	x	x	1	W S1: CONTROL	*
1	x	x	1	R S1: STATUS	*
0	X	0	1	R/W S0 (data)	*
0	x	1	1	R/W S3 (interrupt vector)	
x	0	x	0	R S3 (interrupt vector ACK cycle)	
x	1	x	0	long-distance mode	

Transmitting data

Before data can be sent via the I²C bus, this must be initialized, and have a state corresponding to a start condition. Bus initialization is achieved by making bit ES0 '1'. The interface can be switched off again by making this bit '0' again. The serial channel may be switched off only after a stop condition has been generated. If this rule is not observed, the controller loses track of the bus operations, and can be synchronized only by a hardware reset.

Before a start condition is generated, the controller finds out if the bus is free by checking if $\overline{BB}$ is '1'. The start condition proper is not generated until the STA bit in the control register is set. Once this is so, an address is sent, complete with a R/$\overline{W}$ bit. The byte transmitted at this stage must be contained in register S0 when the STA bit is set. After transmitting the device address, the data exchange is initiated.

A repeated start condition is generated in a slightly different manner than a normal start condition. After the bus has been occupied, setting the STA bit has no effect. A start condition is generated, and data is transmitted from S0, only after data is sent to S0. A stop condition is necessary to mark the end of a transmission. This is achieved by setting the STO bit, and writing a value to S0. The actual value written to S0 is irrelevant, since it is ignored.

The transfer of data may commence after the start condition has been set up, and the address, complete with a R/$\overline{W}$ bit, has been sent. The moment the transmission starts, the PIN (pending interrupt) bit is set. This bit is reset automatically at the end of the transmission. Provided the ENI (enable interrupt) bit had been made '1' by the user, the $\overline{INT}$ output is triggered. The interface card described here does not use the interrupt pulse. Instead, the control software 'polls' the PIN bit. Every polling action requires waiting for the PIN bit to revert to '0' again.

The state of the LRB (last received bit) indicates whether or not a salve has returned an acknowledge condition to indicate correct reception of the data. As soon as PIN is '0' again, the data to be transmitted may be put into register S0. Next, PIN becomes '1' again, the data is transmitted, and PIN becomes '0'. The LRB again indicates that an acknowledge has been generated, 'which means the data has arrived securely at the slave address. This sequence is repeated as long as the master sends data. The end of the sequence may be marked by a stop condition to indicate that the bus is free, and accessible to other devices. Since the end of the stop condition can not be detected by looking at the PIN bit, the software must wait until $\overline{BB}$ reverts to '1'.

When receiving data, the master must generate an acknowledge condition (i.e., set the ACK bit) after every received byte. The addressing of the master is similar to the transmitting of data. After addressing, the PIN bit is '0', so that the IC functions as a master/receiver. The controller keeps the SCL line low until the PIN bit is set by a read operation to register S0. This is an indication for the slave/transmitter that the master is ready to receive data. This means that S0 must be read once before the reception of data can commence. The value read from S0 to initiate reception has no significance. The IC subsequently awaits data, generates an acknowledge, and resets the PIN bit. After that, the data may be read from register S0, which resets the PIN, and prepares the IC for reception of the next byte.

Once the desired data has been received, the ACK bit must be reset before S0 can be read. The slave will send one more byte, of which the reception is not acknowledged by the master. After that, the communication may be ended with the aid of a stop condition. The consequence of this protocol is that the slave will transmit at least one byte too many.

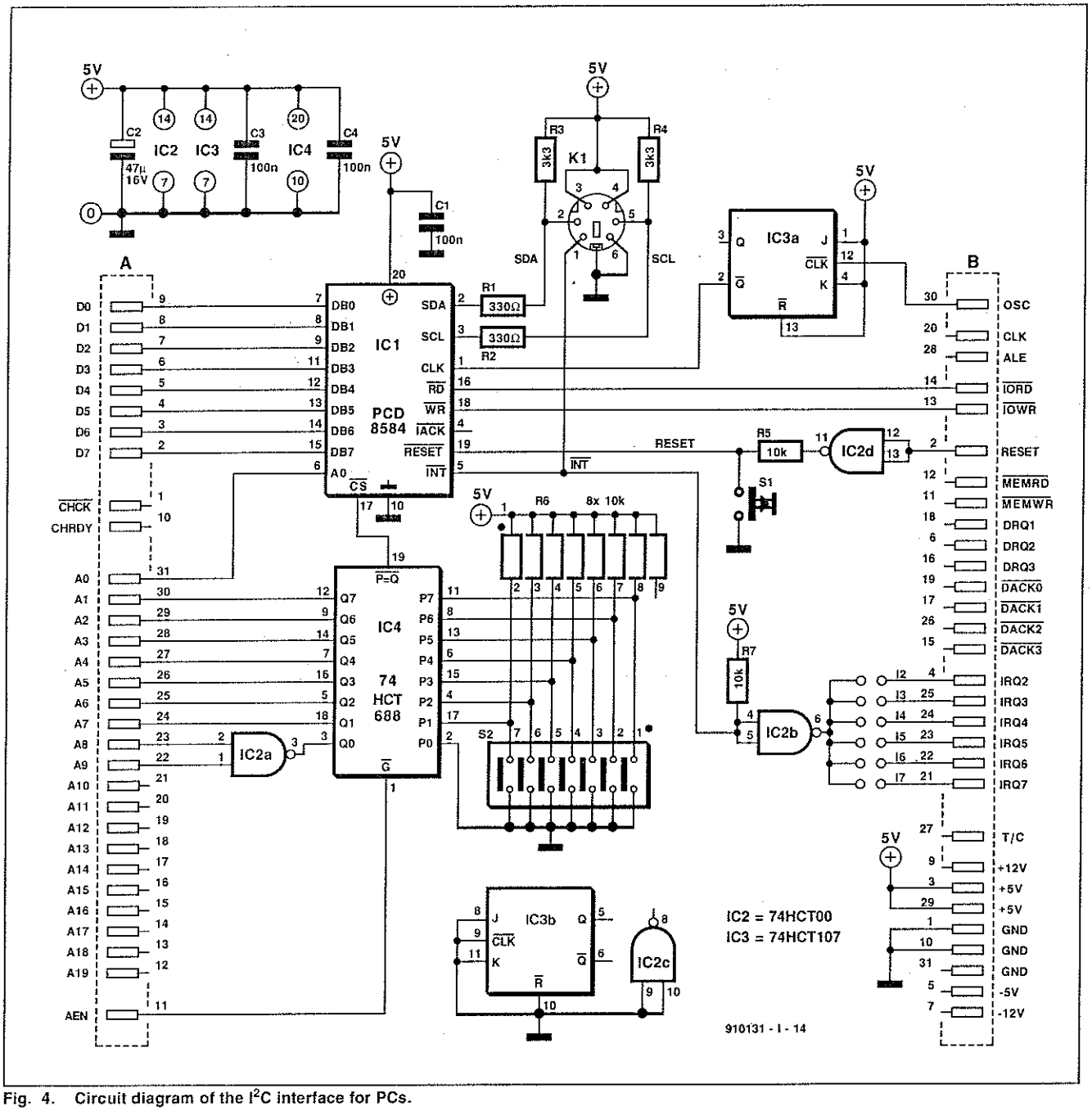


Fig. 4. Circuit diagram of the I²C interface for PCs.

its (mostly passive) 'monitor mode'. On the interface card, this register is of no importance because the PCD8584 is the only master device on the I²C bus, and thus automatically assumes the function of co-ordinating all bus actions.

S1, control/status register. This register is available double. It is addressed when pin A0 is high. All other registers are addressed when pin A0 is low. The selection of the latter depends on bits ES0, ES1 and ES2 in the register 'S1: control'. The selection between control and status register is effected with the aid of the ES0 bit: as long as this is '0', only the control register is accessible for reading from or writing to. Also, the serial interface is switched off. When ES0 is '1', 'S1: control' is written to, 'S1: status' is read from, and the serial interface is actuated. The remaining bits are not involved until they are significant for the control software.

S2, clock register. The clock pulses on the SCL line are derived from the signal at the CLK input. Bits S20 and S21 in register S2 allow one of four bus clock frequencies (SCL) to be selected: 1.5 kHz, 11 kHz, 45 kHz or 90 kHz. The bits S22, S23 and S24 are used to select the frequency at the clock input of the IC: 3 MHz, 4.43 MHz, 6 MHz, 8 MHz or 12 MHz. The latter is the default after a reset.

S3, interrupt vector. When the controller is used on interrupt basis, it is capable of putting an interrupt vector on the PC bus. This happens when input $\overline{IACK}$ is made low. Since this function is not used on the PC interface card, S3 has no function in the present application.

Around the controller

In addition to the PCD8584, three other IC and a handful of passive components are required to build the PC interface (see the circuit diagram in Fig. 4).

Circuits IC4 and IC2A form the address decoder, which compares the address on the PC bus with that set on switch block S2. When the two addresses match, the $\overline{CS}$ line to the PCD8584 is pulled low, so that a read or write command in the I/O range finds its way to the I²C controller. Address signal A0 goes directly to the IC, and needs no further treatment. Since in this case the PC bus is loaded with only one IC input, additional buffering is not required. The $\overline{IORD}$ and $\overline{IOWR}$ lines are connected directly to the $\overline{RD}$ and $\overline{WR}$ inputs of the IC. The interrupt out-

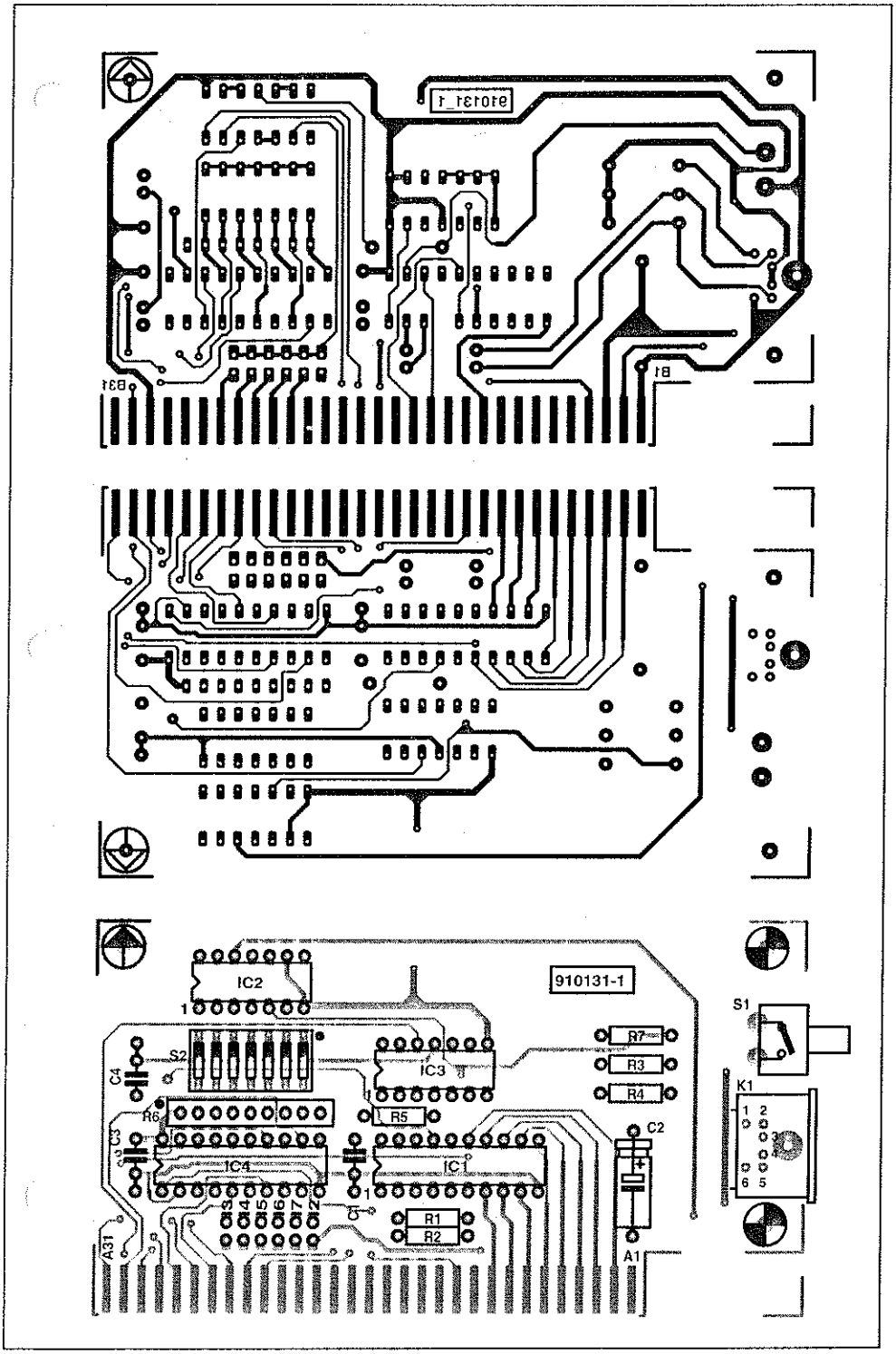


Fig. 5. The printed circuit board for the interface is double-sided and through-plated.

put signal is inverted and buffered by IC2b. Pull-up resistor R7 and connector K1 allow the (open drain) $\overline{INT}$ outputs of further I²C ICs to be taken up into an interrupt chain with an OR function. A jumper is used to select the interrupt line in the PC. The jumper is not fitted unless interrupts are required (the control software described further on does not use interrupts generated by the interface).

The output of gate IC2d is made short-circuit resistant by resistor R5. This is done to allow the circuit to be reset by the PC as well as manually with the aid of a push-button, S1. This is particularly useful while experimenting, since a soft reset (CTRL-ALT-DEL) on the PC does not actuate the reset line on the bus expansion slot.

Bistable IC3a divides the 14.3-MHz bus clock signal by two. This provides an output clock of 7.16 MHz, which can be handled without problems by the controller when programmed to operate in 8-MHz mode.

Connector K1 is a mini-DIN type on which the SDA, SCL and $\overline{INT}$ lines are available, and, of course, ground and +5 V. This allows ready connection of the present interface to further (experimental) I²C systems. Resistors R1, R2, R3 and R4 give SCL and SDA the optimum line impedance.

Control software

The control software developed for the present interface will be discussed in detail in relation to our first I²C application circuit, an A/D-D/A converter, to be published next month. The software enables the I²C interface to be controlled just as 'easily' as, say, a printer or a graphics card. The driver written for the I²C interface relieves the user of observing bus protocols—all this can be left safely to the software and the bus controller. Interestingly for the programmers among you, the I²C interface driver is supplied with a commented source listing.

COMPONENTS LIST

Resistors:

2	330Ω	R1;R2
2	3kΩ3	R3;R4
2	10kΩ	R5;R7
1	8-way 10kΩ array	R6

Capacitors:

3	100nF	C1;C3;C4
1	47µF 16V	C2

Semiconductors:

1	PCD8584	IC1
1	74HCT00	IC2
1	74HCT107	IC3
1	74HCT688	IC4

Miscellaneous:

1	Push-to-make button with angled terminals	S1
1	7-way DIL switch	S2
1	6-way PCB-mount mini DIN socket	K1
1	Printed circuit board	910131-1

Construction

The printed-circuit board (Fig. 5) is a double-sided, through-plated type. The ready-made PCB supplied through the Readers Services has gold-plated PC slot fingers. The construction is straightforward as only relatively few parts are used. As customary with PC extension cards, the board is secured to a support bracket. This is drilled such that the I²C extension connector and the reset switch are accessible on the rear panel of the PC. As it will seldom be required to reset the I²C controller, you may want to omit S1.

The mini-DIN socket carries all the signals required to connect further I²C ICs to the bus.

The DIP switches contained in S2 are set to the desired I/O base address, which lies in the range from 300_H to $3FE_H$. Note that the range 300_H to $31F_H$ is actually reserved for experimental cards. Other ranges may be used, however: for instance, $3D0_H$ to $3DF_H$ if you do not have a CGA card in your PC. When the I²C control software is loaded, the interface address is communicated to the driver proper. This arrangement should enable a suitable address block to be found for the card in almost any PC. ■

Reference:

1. "Inter-IC communications", *Elektor Electronics* September 1990.