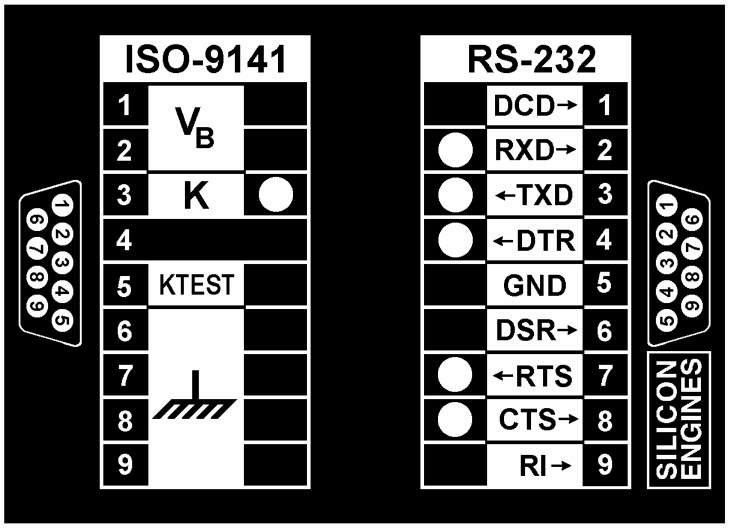
9141 CONVERTER, MODEL 9001

USER'S GUIDE



SILICON ENGINES
2101 OXFORD ROAD
DES PLAINES, IL 60018
847-803-6860
FAX 847-803-6870
www.siliconengines-ltd.com

1. OVERVIEW

1.1. INTRODUCTION

This document provides installation instructions for the Silicon Engines **9141 Converter** module, Model 9001—a compact electronic device that allows a personal computer to connect to an automotive diagnostic data link. This module is compatible with ISO-9141, a world-wide standard issued by the International Standards Organization.

The ISO-9141 Converter is also frequently used to support Keyword Protocol 2000 (ISO-14230), an expanded version of ISO-9141.

1.2. APPLICATIONS

- **Development:** Facilitates development of an automotive ECU (electronic control unit) that supports an ISO-9141 diagnostic line, by allowing a personal computer to act as the diagnostic analyzer during software development.

- **Production:** Allows the ISO-9141 data link to serve as a port for testing the ECU, and for downloading constants and parameters: serial number, calibration data, etc.

- **Service:** For concept cars, prototypes, and low-volume vehicles, allows a PC to act as a diagnostic analyzer.

1.3. FUNCTIONS

- **Level conversion:** Converts signals between ISO-9141 levels and RS-232 (CCITT V.24) levels, for connection to a personal computer.

- **Speed switching:** Contains logic for shifting between 5-baud and high speed operation, facilitating PC interfacing.

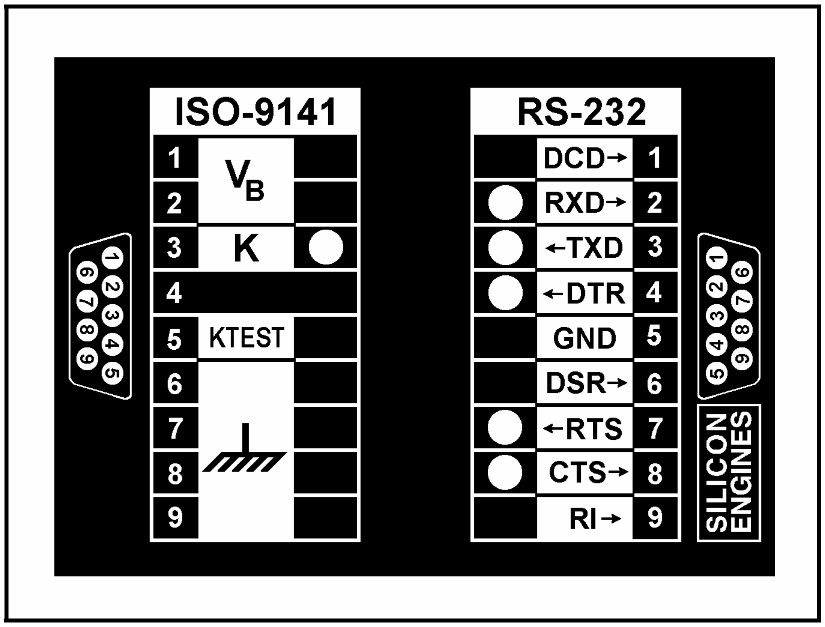
- **Duplexing:** Converts the half-duplex ISO-9141 line to full-duplex RS-232 signals, in association with available PC software.

- **Signal indicators:** Provides six two-color LEDs to show the states of all significant signal lines.

2. ENCLOSURE

2.1. TOP PANEL

The 9141 Converter is housed in a black plastic enclosure. Six LED indicator lamps appear through clear plastic windows on the top panel. The legends on the top panel identify the functions of these lamps, and identify the signals on the ISO-9141 and RS-232 connectors.



9141 CONVERTER TOP PANEL
FIGURE 2.1.1.

2.2. ENCLOSURE SIZE

WIDTH	HEIGHT	DEPTH
4.375 IN	3.25 IN	1.5 IN
111 MM	82,6 MM	38,1 MM

ENCLOSURE DIMENSIONS
FIGURE 2.2.1.

3. CONNECTORS

3.1. ISO-9141 CONNECTOR

The connector at the left of the 9141 Converter is a type DB9M plug (9-pin male D sub-miniature).

A DB9F (female DB9) socket plugs in here. Three signal wires are required: VBATT, the ISO-9141 K-line, and ground.

PIN NO.	SYMBOL	SIGNAL	DESCRIPTION
1-2	V_B	VBATT	+12 VOLT BATTERY POWER
3	**K**	K	ISO-9141 K-LINE
4	■	NC	*NO CONNECTION*
5	**KTEST**	KTEST	TTL LEVEL K OUTPUT
6-9	⏚	GROUND	POWER AND SIGNAL RETURN

ISO-9141 CONNECTOR PIN-OUTS
FIGURE 3.1.1.

These pin-outs, as well as the locations of the pins within the 9-pin connector, are shown on the 9141 Converter top panel legend *(Fig. 2.1.1)*.

Because the connectors vary on each automotive ECU, the user must construct his own cable to connect to the 9141 Converter. Or contact Silicon Engines for assistance.

At the ISO-9141 Converter side, use a DB9F connector. Connect VBATT to pin 1, the K-line to pin 3, and ground to pin 6. The locations of the pins within the connector are shown on the top panel decal *(see Fig. 2.1.1). (See part 4 below for more information on VBATT and GND.)*

The KTEST line provides a TTL-level (5 volts, ground) image of the ISO-9141 K-line, with a high-impedance (100 KΩ) output, useful for connection to test equipment.

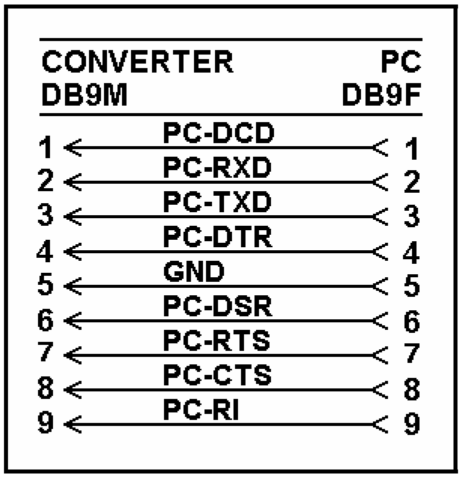
3.2. RS-232 CONNECTOR

The connector at the right of the 9141 Converter is a DB9F type, designed for connection to the COM1 port of a PC. Signal wires are shown on the top panel artwork *(Fig 2.1.1)*.

Connector polarities and signal names have been chosen for compatibility with the RS-232 (CCITT V.24/V.28) serial port of a recent model IBM®-compatible PC. PC-AT and later computer models typically provide a DB9M connector at their COM1 serial port.

Typically the 9141 Converter connects to the PC over a DB9 extender cable about 6 feet (2 M) long. A DB9M connector plugs into the 9141 Converter at one end of the cable, and a DB9F connector plugs into the PC at the other end.

This cable should provide straight-through wiring: pin 1 connects to pin 1, pin 2 to pin 2, etc. One each of this cable is provided with each Model 9001 9141 Converter.



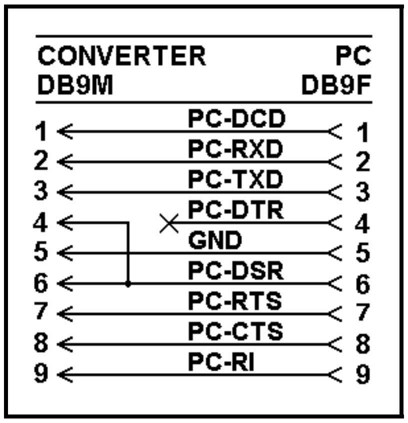
STRAIGHT-THROUGH RS-232 CABLE
FIGURE 3.2.1.

3.3. SPECIAL CABLE FOR DELPHI CUSTOMERS

Delphi Automotive provides special diagnostic software packages to its customers for use in servicing automotive ECUs developed and manufactured by Delphi. For the convenience of Delphi customers, Silicon Engines offers a special RS-232 cable for these applications.

For Delphi applications, please specify 9141 Converter Model 9001-2.

The RS-232 cable provided with our Model 9001-2 is wired as shown below. This cable provides a loopback connection between pins 4 and 6 on the 9141 Converter side. This loopback compensates for the fact that certain versions of the Delphi software do not control the signal DTR, which the 9141 Converter needs to operate in high-speed data mode.



SPECIAL DELPHI CABLE PROVIDED WITH MODEL 9001-2
FIGURE 3.3.1.

3.4. 25-PIN SERIAL PORT

If your computer provides a 25-pin DB25F RS-232 serial port connector, it will be necessary to install an adapter with a DB25M plug on one side, and a DB9M plug on the other, to adapt the computer to the 9141 Converter. This type of adapter is commonly sold to adapt serial mouse devices to the 25-pin ports, and should be readily available from computer stores—or contact Silicon Engines for assistance.

4. POWER REQUIREMENTS

4.1. CONNECTING VBATT AND GROUND

When working with an ECU at the component level, the VBATT and GROUND lines are typically connected (along with the K-line) to a suitable connector on the ECU. Both the 9141 Converter and the ECU are powered from the vehicle's +12 volt battery.

The 9141 Converter can also be powered by a +12 VDC power supply that connects to building AC power lines.

4.2. POWER LEVELS

The 9141 Converter contains built-in power supply circuitry that generates needed power from VBATT and GND. The unit produces internal +5 VDC for digital logic and the LED indicators, as well as ±10 VDC for the RS-232 interface.

SPECIFICATION	MIN.	TYP.	MAX	UNITS	CONDITIONS
SUPPLY VOLTAGE VBATT	9.0	13.8	16.0	VDC	CONTINUOUS OPERATION
			24.0	VDC	DOUBLE BATTERY, 1 MINUTE MAX.
			60	VDC	LOAD DUMP, 100 MS MAX.
			-24	VDC	REVERSE BATTERY
			-100	VDC	NEGATIVE TRANSIENTS
SUPPLY CURRENT		65		MA	VBATT=+13.8 VDC, K-LINE IDLE
		82		MA	VBATT=+13.8 VDC, K-LINE LOW

SUPPLY POWER SPECIFICATIONS
FIGURE 4.2.1.

4.3. LOAD DUMP PROTECTION

The 9141 Converter contains circuitry for protection against automotive load dump transients up to the maximum level shown above. These levels are adequate for most current vehicle designs.

However, if higher transient levels are anticipated, measures should be taken to protect the 9141 Converter. One method is to power the device from an AC line-powered +12 volt power supply, rather than from the vehicle's battery.

4.4. SEPARATE VBATT SOURCES

If the ECU and the 9141 Converter are powered from separate sources, (a) the ground of the 9141 Converter must be connected to the ground of the ECU, and (b) the level of VBATT provided to the 9141 Converter should be within ±3 VDC of the level of VBATT provided to the ECU.

4.5. DOUBLE-BATTERY PROTECTION

The 9141 Converter is capable of withstanding temporary connection to a double-battery jump start (+24 volt input), but should not be operated outside of the continuous operation limits shown above for more than a short period, or damage to the device may result.

4.6. REVERSE BATTERY PROTECTION

The 9141 Converter is protected against inadvertent reverse battery connection. The unit will not operate properly with reversed power inputs, but will not be damaged.

5. LAMP FUNCTIONS AND SIGNAL FLOW

5.1. TWO-COLOR LAMPS

Six LED indicators are visible on the top panel of the 9141 Converter. Each is a two-color device that glows either green or red whenever the device is powered up.

LED COLOR	SIGNAL LINE CONDITION
GREEN	HIGH VOLTAGE LEVEL
RED	LOW VOLTAGE LEVEL

LED COLOR CODES
FIGURE 5.1.1.

5.2. SIGNAL FUNCTIONS

One LED indicator shows the voltage level on the ISO-9141 K-line.

CONN. PIN	SIGNAL NAME	DATA DIR.	LAMP COLOR	TYP. LEVEL	SIGNAL FUNCTION
3	K	ECU↔PC	GREEN	+12 V	IDLE LINE; LOGIC 1(MARK); STOP BIT
			RED	0 V	START BIT; LOGIC 0 (SPACE)

ISO-9141 K-LINE
FIGURE 5.2.1.

Five LED indicators show the voltage levels on key RS-232 signal lines.

CONN. PIN	SIGNAL NAME	DATA DIR.	LAMP COLOR	TYP. LEVEL	SIGNAL FUNCTION
1	DCD	ECU→PC	*(NONE)*	+10 V	DATA CARRIER DETECT, HIGH WHEN POWER IS ON
2	RXD	ECU→PC	GREEN	+10 V	*HIGH SPEED*: START BIT; LOGIC 0 (SPACE)
			RED	-10 V	*HIGH SPEED*: IDLE LINE; LOGIC 1 (MARK); STOP BIT
3	TXD	ECU←PC	GREEN	+10 V	*HIGH SPEED*: START BIT; LOGIC 0 (SPACE)
			RED	-10 V	*HIGH SPEED*: IDLE LINE; LOGIC 1 (MARK); STOP BIT
4	DTR	ECU←PC	GREEN	+10 V	ON=HIGH SPEED DATA MODE
			RED	-10 V	OFF=5 BAUD SET-UP MODE
5	GND	---	*(NONE)*	0 V	SIGNAL COMMON
6	DSR	ECU→PC	*(NONE)*	+10 V	DATA SET READY, HIGH WHEN POWER IS ON
7	RTS	ECU←PC	RED	-10 V	5 *BAUD*: IDLE LINE; LOGIC 1 (MARK); STOP BIT
			GREEN	+10 V	5 *BAUD*: START BIT; LOGIC 0 (SPACE)
8	CTS	ECU→PC	RED	-10 V	5 *BAUD*: IDLE LINE; LOGIC 1 (MARK); STOP BIT
			GREEN	+10 V	5 *BAUD*: START BIT; LOGIC 0 (SPACE)
9	RI		(NONE)	0 V	RING INDICATOR, NOT SUPPORTED

RS-232 SIGNALS
FIGURE 5.2.2.

5.3. SELECTING BETWEEN 5 BAUD AND HIGH-SPEED OPERATION

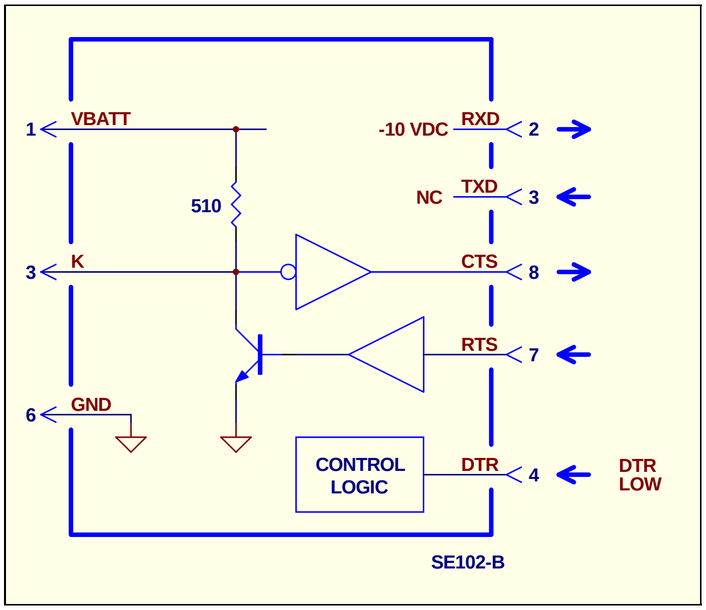
The PC uses its DTR (Data Terminal Ready) line to select between two modes of operation. When DTR is low, the 9141 Converter is switched to 5 baud mode. Signals are exchanged at 5 bits per second between the PC and the ECU, to start up ISO-9141 communications.

After suitable 5 baud signals are exchanged, the PC turns on DTR, switching the 9141 Converter to high-speed operation. Typical speeds are 600, 4800, or 10,417 bits per second. The high-speed rate is under software control.

5.4. LOW-SPEED (5 BAUD) MODE

When the 9141 Converter is in low-speed mode (DTR low), the RS-232 control signals RTS and CTS are used for 5-baud communications with the ECU. Software within the PC generates the 5 baud ISO-9141 initialization sequence. A high level input on the RTS line will cause the K-line to go low. The state of the K-line is received by the PC on its CTS line.

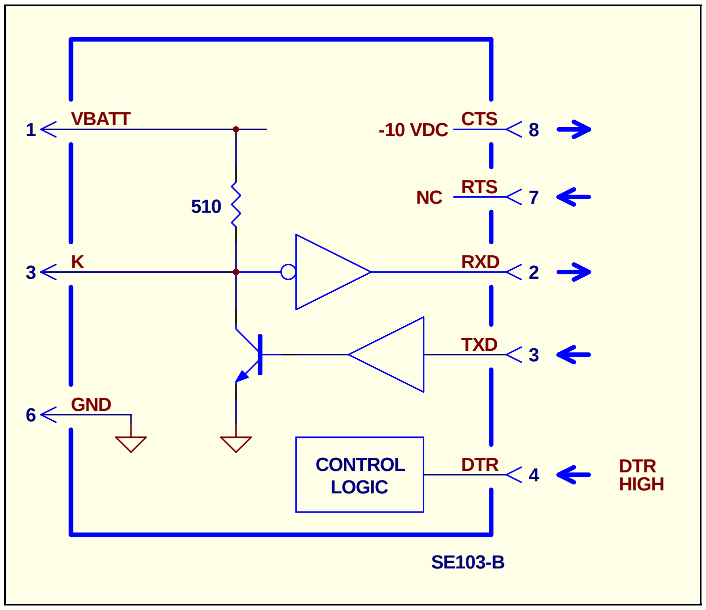
In low-speed mode, the TXD line (the high speed data line from the PC) is locked out, and has no influence on the ISO-9141 K-line. The RXD line is clamped to the marking state (idle line, logic 1, –10 VDC), regardless of the state of the ISO-9141 K-line.



SIGNAL FLOW, 5 BAUD MODE
FIGURE 5.4.1.

5.5. HIGH-SPEED MODE

When the 9141 Converter is in high-speed mode (DTR high), the RTS line (the 5 baud input line from the PC) is locked out, and has no influence on the ISO-9141 K-line. The CTS line is clamped to the OFF state (–10 VDC), regardless of the state of the ISO-9141 K-line.



SIGNAL FLOW, HIGH SPEED MODE
FIGURE 5.5.1.

5.6. LAMP PATTERNS, SYSTEM IDLE

ISO-9141 LAMP				RS-232 LAMPS			
LAMP	**COLOR**	**STATE**	**LEVEL**	**LAMP**	**COLOR**	**STATE**	**LEVEL**
K	GREEN	IDLE	+VBATT	RXD	RED	IDLE	-10 VDC
				TXD	RED	IDLE	-10 TO -15 VDC
				DTR	RED	OFF	-10 TO -15 VDC
				RTS	RED	IDLE	-10 TO -15 VDC
				CTS	RED	IDLE	-10 VDC

LAMP PATTERNS, ISO-9141 SYSTEM IDLE
FIGURE 5.6.1.

The above lamp pattern should appear when the system is idle. The 9141 Converter is connected to both the PC and the ECU, and the power is on. The 9141 software in the PC is *not* running, so the RS-232 signal lines from the PC are all at idle levels. The RS-232 lamps are all red, while the ISO-9141 K lamp is green.

K LAMP STEADY RED: If the K lamp is red when the system is idle, check the ECU and the cables for a short to ground. The K lamp should be green most of the time in any mode. When the system is communicating, the K lamp will be green while the ISO-9141 line is idle waiting for a new command, and during logic 1 data bits, stop bits, and interframe idle time. The K lamp should go red only for start bits and logic 0 data bits. A steady red K lamp indicates a problem on the K line.

5.7. LAMP PATTERNS, 5 BAUD MODE

ISO-9141 LAMP				RS-232 LAMPS			
LAMP	**COLOR**	**STATE**	**LEVEL**	**LAMP**	**COLOR**	**STATE**	**LEVEL**
K	GREEN/RED	1/0	5 BAUD DATA	RXD	RED	IDLE	-10 VDC
				TXD	RED	IDLE	-10 TO -15 VDC
				DTR	RED/GREEN	OFF	-10 TO -15 VDC
				RTS	RED/GREEN	1/0	5 BAUD DATA
				CTS	RED/GREEN	1/0	5 BAUD DATA

5 BAUD LAMP PATTERNS
FIGURE 5.7.1.

During 5 baud ISO-9141 initialization, the K lamp will be green with long red flashes.

When the ECU is transmitting 5 baud data, the CTS lamp (data from ECU to PC) will be red with long green flashes. The RTS line (data from PC to ECU) will be solid red (idle).

When the PC is transmitting 5 baud data, *both* the CTS and the RTS lamps will be red with long green flashes. The 9141 Converter echoes the data it receives from the PC on the RTS line, back to the PC on the CTS line. *(See the signal flow diagram in Fig. 5.4.1.)*

5.8. LAMP PATTERNS, HIGH SPEED MODE

ISO-9141 LAMP				RS-232 LAMPS			
LAMP	**COLOR**	**STATE**	**LEVEL**	**LAMP**	**COLOR**	**STATE**	**LEVEL**
K	GREEN/RED	1/0	HIGH SPEED DATA	RXD	RED/GREEN	1/0	HIGH SPEED DATA
				TXD	RED/GREEN	1/0	HIGH SPEED DATA
				DTR	GREEN	ON	+10 TO +15 VDC
				RTS	RED	IDLE	-10 TO -15 VDC
				CTS	RED	IDLE	-10 VDC

HIGH SPEED LAMP PATTERNS
FIGURE 5.8.1.

During high speed mode, the K lamp will be green with short red flashes.

When the ECU is transmitting high-speed data, the RXD lamp (data from ECU to PC) will be red with short green flashes. The TXD line (data from PC to ECU) will be solid red (idle).

When the PC is transmitting high speed data, *both* the RXD and the TXD lamps will be red with short green flashes. The 9141 Converter echoes the data it receives from the PC on the TXD line, back to the PC on the RXD line. *(See the signal flow diagram in Fig. 5.5.1.)*

6. COMPUTER SOFTWARE

6.1. SOFTWARE FUNCTIONS REQUIRED

The 9141 Converter is designed for operation in conjunction with special software that runs on the attached personal computer. This software carries out ISO-9141 5 baud initialization; handles echoed characters from the half-duplex unit link; switches the 9141 Converter between 5 baud mode and high speed mode; and then carries out high speed communications.

Different software versions are required to support ISO-9141-CARB (emission control) and other versions of the ISO-9141 protocol used by various vehicle manufacturers (ECU diagnostics).

The 9141 Converter is also frequently used to support Keyword Protocol 2000 (ISO-14230), an expanded version of ISO-9141 used primarily by European automotive manufacturers. For further information, contact Silicon Engines.

6.2. APPLICATION NOTES

Please see Appendix A below for suggestions on the PC software required to run the 9141 Converter.

Please contact Silicon Engines if you need engineering assistance with your 9141 application.

APPENDIX A. HOW TO DEVELOP ISO-9141 APPLICATIONS IN C

A.1. TOOLS USED

This Application Note applies to ISO-9141 systems which use a single K-line, supporting half-duplex serial communications. The Silicon Engines 9141 Converter allows a PC to use a serial port and serial communications software to communicate with the ISO-9141 device.

Three software packages are needed in order to write the application: a C compiler, a serial communications handler, and a timer library (if your application requires fast initialization).

A.2. C COMPILER

The leading brands of C compilers for the IBM-PC are Microsoft and Borland. Other varieties of C compilers exist for the IBM-PC, but you have to be careful that the libraries that you buy (see below) are compatible with those compilers. At Silicon Engines, we use Borland C/C++ 4.02, but any version of C/C++ including Builder will work. Note that Borland C/C++ 4.5 and lower only work up to the year 2038 as far as time/date operations go. Contact Borland at: http://www.borland.com.

A.3. COMMUNICATIONS HANDLER OPTIONS

There are three communications options that we have worked with at Silicon Engines.

- **BIOS only:** First there is the option of not using a COMM library and just programming using BIOS commands. This is not recommended. However if you do not plan to use a third-party COMM library, you can use the BIOS directly. It is recommended that you buy a book about RS-232 programming using the IBM-PC BIOS, such as Joe Campbell's *C Programmer's Guide to Serial Communications* (Howard W. Sams & Company, 1989) or Timothy S. Monk's *Windows Programmer's Guide to Serial Communications* (Sams Publishing, 1992). If your PC has an 8250, 16540, or 16550 UART or serial communications IC—which it probably does—then you can use the following code segments to access the DTR and RTS lines, for example:

```
#define COMPORT (1) /* change this to 2 for COM2 or 3 for COM3, etc. */
#define BASEADDR (0x3F8) /* change this to 2F8 for COM2, 3E8 for COM3, etc. */

#define MCR *(unsigned char *)(BASEADDR+4)

#define DTRON MCR|=1 /* this sets DTR on */
#define DTROFF MCR&=~1 /* this turns DTR off */

#define RTSON MCR|=2 /* this sets RTS on */
#define RTSOFF MCR&=~2 /* this turns RTS off */

#define IRQON MCR|=8 /* this enables serial interrupts */
#define IRQOFF MCR&=~8 /* this disables serial interrupts */
```

For setting 5 baud and 10400 baud using normal BIOS routines (i.e. using TXD and RXD, not the special RTS/CTS DTR off mode available on the Silicon Engines 9141 Converter), use the following formula:

```
divisor = 1843200 / (16 * baud_rate)
```

For 5 and 10400, the values for *divisor* are:

```
#define BAUD_5_HIGH (90)
#define BAUD_5_LOW (0)
#define BAUD_10400_HIGH (0)
#define BAUD_10400_LOW (11)
```

The code that pokes the baud rate into the UART is:

```
#define MSB *(unsigned char *)(BASEADDR+0)
#define LSB *(unsigned char *)(BASEADDR+1)
#define LCR *(unsigned char *)(BASEADDR+3)

#ifdef DO_5_BAUD
#define SET_BAUD_RATE { LCR=0x80; MSB=BAUD_5_HIGH;

LSB=BAUD_5_LOW; LCR=7; }
#else
#define SET_BAUD_RATE { LCR=0x80; MSB=BAUD_10400_HIGH; \

LSB=BAUD_10400_LOW; LCR=3; }
#endif
```

- **WCSC COMMDRV library:** The main disadvantage to using this library is that it cannot run at very high serial speeds (115,200 baud) very well (as of the 1992 version—later versions may have fixed this limitation). The advantages are that it allows for special-purpose handlers based on specific receive or transmit events, and also allows for transmitting bytes immediately for precision timing, which can be used for fast initialization. This software is available at http://www.wcscnet.com.

Here is the code for fast initialization using WCSC's COMMLIB:

```
#define BAUD_DIV(x) ((unsigned short)(1843200L/(16L*((long)(x)))))

unsigned short far CheckSendnowDone(short val, struct port_param far *p)
{
    CheckSendnowDoneFlag=1;
    if ((val>>8)==INTFUNC_VSENDNOWDONE) {
        SendnowDoneFlag=1;
    }
}

unsigned short (far *func)(short v,struct port_param far *p)=CheckSendnowDone;
int InitComm(int port,int subport,unsigned short addr,unsigned short irq,
    unsigned short cardtype,unsigned short cardseg,
    unsigned short inbuflen,unsigned short outbuflen,
    unsigned short flag,unsigned addrtype)
{
int x;
char *buf;
struct port_param far *pcb;
struct { unsigned count; unsigned char data[1]; } nsp;
```

```
/*** THE FOLLOWING INITIALIZES THE PORT ***/

x=inbuflen+outbuflen+2;

if ((buf=(char *)malloc(x))==NULL) AllocError(x);
x=sizeof(struct port_param);
if ((pcb=(struct port_param *)malloc(x))==NULL) AllocError(x);
while (x--) ((char*)pcb)[x]=(char)0;
if ((x=ser_rs232_getport(port,pcb))) CommDrvError(x);

pcb->ser_rs232_base= addr;
pcb->irq = irq;
pcb->parity = PARITY_NONE;
pcb->brk = BREAK_OFF;
pcb->lngth = LENGTH_8;
pcb->stopbit = STOPBIT_2;
pcb->block[0] = 2;
pcb->hblock[0] = 0;
pcb->block[1] = 2;
pcb->hblock[1] = 0;
pcb->protocol = PROT_NONNON;
pcb->buffer_seg = (unsigned short)(((long)((void far

*)buf))>>16);
pcb->buffer_off = (unsigned short)((long)(void far *)buf);
pcb->inbuf_len = inbuflen;
pcb->outbuf_len = outbuflen;
pcb->cardtype = cardtype;
pcb->aux_addr1 = subport;
pcb->flag2 = 0 /* CHAIN_INT */ ;
pcb->sflag = flag;
pcb->cardseg = cardseg;
if ((x=ser_rs232_setup(port,pcb))) CommDrvError(x);
ser_rs232_rts_off(port);
ser_rs232_dtr_on(port);

/*** THE FOLLOWING DOES THE FAST INITIALIZATION ***/

if ((x=ser_rs232_getport(comport,pcb))) CommDrvError(x);
pcb->lngth=LENGTH_8; pcb->parity=PARITY_NONE; pcb->stopbit=STOPBIT_1;

pcb->block[0]=0; pcb->hblock[0]=0;
pcb->block[1]=0; pcb->hblock[1]=0; /* 18.2*2 ms > 30 ms */
pcb->baud=BAUDUSER02;

if ((x=ser_rs232_setbauddiv(card,(pcb->baud=BAUDUSER02), BAUD_DIV(360))))
CommDrvError(x);
if ((x=ser_rs232_setup(comport,pcb))) CommDrvError(x);
nsp.count=1; nsp.data[0]=0x00;
CheckSendnowDoneFlag=0; SendnowDoneFlag=0;
ser_rs232_set_intfunc(comport,&func,INTFUNC_OTHER);
ser_rs232_misc_func(comport,SENDNOW,(long)(&nsp));
while (!SendnowDoneFlag) {
          if (CheckSendnowDoneFlag) {
          if (!CheckNowDoneFlag) Commfailure();
          }
    }
```

```
/***Wakeup signal sent, now go to fast baud rate and send Start Communications
message.***/
```

- **Greenleaf library:** The Greenleaf library has many different drivers for all sorts of different setups, plus a fast driver for 115,200 applications. It does not have the SENDNOW function for immediately sending a byte, which makes fast init difficult.

· Greenleaf has a website at http://www.gleaf.com.

- **Recommendation:** At Silicon Engines, we use the WCSC Library for ISO-9141 applications, and the Greenleaf library for most other serial communications applications.

  However, fast init is possible using Greenleaf (and a timer library) by setting DTR LOW and setting RTS low for 25 milliseconds and then high for 25 milliseconds—as discussed further below.

A.4. TIMER LIBRARIES

At Silicon Engines, we have experimented with two different timer libraries for handling functions that can do timing of millisecond and even microsecond resolution. This is necessary only for fast initialization. Without a special timer library, the only resolution you will get from BIOS is 55 msec (1/18 of a second). The libraries investigated are the Precision Zen Library, and the PC Timer Tools Library version 5.0.

- **Precision Zen:** The main advantage to this library is that it is a public domain program. It should be available from shareware software providers. The only problem with the version we have is that it doesn't work properly on some systems. It is recommended for professional use to use a different library.

- **PC Timer Tools:** This library works great on all the systems we have tried it on. It allows millisecond and microsecond timers for all your precision timing needs. PC Timer Tools 5.0 (or PCHRT) is available from Ryle Design at http://www.ryledesign.com.

A.5. 9141 INITIALIZATION

There are two different methods of initializing (waking up) an ISO-9141 device. There is slow init (at 5 baud) and fast init (50 millisecond signal).

- **Slow initialization (5 baud):** One method is to use TXD and RXD within the PC's serial communications channel, and set the PC to 5 baud. This is done using a user-defined baud rate. Depending on your COMM library, this is done in different manners (see above). You send the address of the module at 5 baud with DTR HIGH.

  Alternately, you can set DTR LOW and use the RTS signal as a data line. The following steps show you how to do this:

  1. Set DTR LOW.
  2. Set RTS HIGH (for START bit).
  3. Wait 200 milliseconds (one bit time at 5 baud).
  4. For each of the eight data bits that you want to send, least significant bit first, set RTS HIGH to send a 0 data bit, or set RTS LOW to send a 1 data bit, and then wait 200 msec.
  5. Set RTS LOW (for STOP bit).
  6. Wait 400 milliseconds (two STOP bit times).
  7. Set DTR HIGH (back to normal communications mode).

- **Fast initialization:** In fast init, you need to bring the K-line down for 25 msec, and then bring it high for 25 milliseconds. This can be done using either the DTR high method—for example, with the WCSC COMMDRV library (see above) —or the DTR low method, as follows:

  1. Set DTR LOW.
  2. Set RTS HIGH.
  3. Wait 25 milliseconds.
  4. Set RTS LOW.
  5. Wait 25 milliseconds.
  6. Set DTR HIGH.
  7. Flush the serial receive buffer, and set the communications IC for the desired high-speed baud rate.
  8. Send the Start Communications token.

A.6. HIGH SPEED COMMUNICATIONS

Since communication using the 9141 Converter is half-duplex, the PC program must be prepared to receive the echo of every byte that it sends. One way to do this with the WCSC library is to use the SENDNOW function to send characters, and then immediately wait for the echo to come back in the transmit byte routine. Other implementations are possible, but keep in mind that most COMM libraries don't immediately transmit a byte, so that there will be a slight delay after sending each byte before the echo is received.

A.7. APPLICATIONS ASSISTANCE

Contact Silicon Engines.

☑