RDX - Reflective Data Exchange

Function Library for Microsoft C

This package implements a very high speed point-to-point communication link between two computers. A 8255 digital i/o chip is used on both computers to implement a digital data transfer, all 24 ports are used. The computers are connected using a 24 wire ribbon cable. The key feature is that no interrupts are required. Also, the system clock is not used so there is no other hardware interference.

These routines use a polling algorithm to perform the transfer. When used with the TASK package, you can easily implement a program that receives data from TEMPO while performing other activities.

The RDX library is a collection of Microsoft 16/32 bit C routines that allow you to receive a stream of characters from a TEMPO server over a high speed digital i/o (ribbon) cable using digital i/o (e.g.: pci-dio24, 48 and 96 and cio-dio24, 48, 96 and 192) cards. RDX only requires an 8255 digital i/o chip at both ends for communication so most cards with this chip will work.

When you write your stimulus program, you can include the RDX library and retrieve characters from your TEMPO protocol. In your protocol, you use dsend() and dsendf() to format a string to send to your stimulus program. In your stimulus program, you call RDX C functions to retrieve the characters from the TEMPO server. RDX works with DOS, Windows 3.1, 95, 98, NT, 2000 and XP.

With Windows NT, 2000 and XP, you have to load a special driver called GIVEIO.

The RDX library requires the base address of the 8255 chip that you will use receive characters from the TEMPO server. For PCI cards, you must use some other method for obtaining the base address such as the Measurement Computing's (formerly Computer Board's) Universal Library.

The "commands" that you send with dsend()/dsendf() are of your own format and creation. It is your stimulus program that receives the character stream and must parse (understand and interpret) it.

Advantages and Limitations of the RDX Library

As with any software library, there are advantages and disadvantages to using RDX. Some advantages are:

1. It is very fast. You can generally transmit bursts up to about 60000 bytes per second, depending on the speed of the computer.
2. It does not use interrupts which means that it does not interfere with the data acquisition on the TEMPO server or stimulus computer.
3. It is a point-to-point, dedicated communication link. Unlike an ethernet connection, no other computer can interfere with (delay) the communication.
4. It is simple but general. It is easy to use with dsendf(). You can create whatever commands you need and easily change them over time if necessary.
5. RDX works with both ISA digital i/o cards and PCI digital i/o cards. With PCI cards, RDX does not provide a way to obtain the base address so you must use some other method for obtaining the base address. Once it is known, you can use the RDX library.
6. Four TTL lines are available for general purpose use on the RDX cable. Two TTLs go from receiver to sender and two TTLs go from sender to receiver (see rdxGetTTL/SetTTL functions.)

7. Your RDX application can runs on Windows 95, 98, 98SE, NT4, 2000 and XP systems.

Some limitations are:

1. It is receive only. If you want to transmit information from your stimulus computer back to the TEMPO server, you must use a different method such as serial or TTL or analog.
2. The maximum distance between the server and stimulus computer are limited to 10-20 feet. This is because the digital i/o ribbon cable should not be longer than this.
3. The TEMPO server currently supports 2 RDX links. So you can control at most two stimulus computers with one server. (If you need more than 2 RDX links, please contact us.)
4. On the receive side, RDX requires that you call an RDX polling function frequently. This often means that your stimulus program will be multithreaded.
5. The RDX library is designed to with Microsoft C 16 bit and 32 bit compilers with DOS or Windows. If you want to run it on a different computer or compiler, contact Reflective Computing.
6. For Windows NT, 2000 and XP, you must install a special driver before running your RDX application.

Nomenclature

In this document, we use the term "Windows XP" to refer to Windows NT, Windows 2000 and Windows XP except where otherwise noted. We use the term "Windows 98" to refer to Windows 95 and Windows 98 except where otherwise noted.

Compiling and Linking

The RDX Library is designed for use with the Microsoft C 16 or 32 bit compilers.

Include RDX.H in your C source code and link with the .OBJ files in the RDX16 or RDX32 directory. The RDX16 directory contains files for 16 bit compilation. The RDX32 directory contains files for Win32 compilation.

For 16 bit applications, compile and link with Large or Huge memory model (compiler switch /AH or /AL).

The .MAK and .LNK files are in each directory. These serve as an example of how to compile and link an application with the RDX Library.

Synopsis of Functions

```
#include "rdx.h"                                          // RDX header file

short dx_open_recv(short nBase, char *pBuf, short nSize); // Open as receiver, return a handle
short dx_getchar(short nHandle, char *pChar);             // Get next received char
short dx_recv(short nHandle);                             // Recv data (returns # chars in buf)
short dx_reset(short nHandle);                            // Reset i/o state & clearbuf
short dx_clearbuf(nHandle);                               // Clear all chars from buffer
short dx_getcount(nHandle);                               // Get # of characters in buffer
short dx_close(nHandle);                                  // Close handle
char *dx_err(Nerr);                                       // Get error message

short dx_GetTTLOUT(short nHandle);                        // Read two RDX TTL outputs
short dx_SetTTLOUT(short nHandle, short n);               // Write two RDX TTL outputs
short dx_GetTTLIN(short nHandle);                         // Read two RDX TTL inputs
```

Description

This package implements a low level protocol for transferring data from one computer to another using one 8255 digital i/o chip. A ring buffer is established to hold data in transit. Once a receiver is opened, you can read characters from the receive buffer.

Data transfer occurs in a polled fashion. The receiver periodically calls dx_recv() which returns the number of characters in its receive buffer (or an error). It performs data transfers with the sender for a limited period of time and returns, essentially immediately, to the caller. At present, TEMPO does not support bi-directional transfers so an alternative means (i.e., serial) must be used to communicate information back to TEMPO.

The RDX Library supports four TTL bits (two in each direction) that can be used by applications for general purpose TTL communication between sender (i.e., your TEMPO protocol on the TEMPO server) and receiver (i.e., your RDX Library application). These TTL bits are independent of RDX communication and do not effect it. The dx_GetTTLOUT(), dx_GetTTLIN() and dxSetTTLOUT() RDX Library functions let you read and write these bits. The RDX TTL bits can be read or written at any time. Reading and writing the RDX TTL bits directly access the digital i/o hardware; any changes take effect at the time the dx_SetTTLOUT() function is called.

RDX Library (Receiver)		**Protocol PCL Function (Sender)**
dx_SetTTLOUT()	----------2 TTL bits---------->	rdxGetTTLIN()
dx_GetTTLIN()	<--------2 TTL bits----------	rdxSetTTLOUT()

Function Return Values

dx_open_recv - Opens receiver. Requires the base address of an 8255 chip and a receive ring buffer. This buffer is not directly accessed by the user. Characters are received using the dx_getchar() function. Note that this buffer should be statically defined as it will be used each time the dx_xxx routines are called. A handle (>0) is returned if no errors occur. Otherwise, an error code is returned.

dx_getchar - Get next received character from the receive buffer. Returns a count of the number of characters in the buffer *before* the next character, if any, is removed from the buffer. Thus, a 0 is returned if

there are no characters. A positive integer is returned if there are (and the character is returned in pChar). A negative number indicates an error code.

dx_recv - Receives data from the communication link. Returns number of characters in buffer. Zero or more characters may be received. This function should be called periodically and frequently. Use dx_getchar to retrieve characters from the receive buffer.

dx_reset - Reset i/o state and clear buffer for the handle. This is normally done once at the beginning of the program but it may be called at other times to force synchronization with the other computer. All data in the buffer is lost. Returns an error code or 0.

dx_clearbuf - Clear all chars in buffer. Returns 0 or an error code.

dx_getcount - Returns the number of characters in buffer.

dx_close - Close handle. Only handles that have been opened by the dx_open _recv should be closed. Once a handle is closed, you may no longer use the handle. Use dx_open_recv to open a new handle.

dx_err - Returns an error message string given a DX error return code. All DX error return codes are negative numbers.

dx_GetTTLOUT - Reads the I/O port and returns the two bits associated with the RDX TTL outputs. These are the two TTLs that the receiver can set and the sender can read. In your TEMPO protocol, see the rdxGetTTLIN() PCL function to read these bits. The return value of dx_GetTTLOUT() is 0, 1, 2 or 3.

dx_SetTTLOUT - Writes the I/O port with the lowest two bits specified in the argument n. The return value is one of the DX_xxx status codes. These bits can be read back with the dx_GetTTLOUT() function. When the bits are set, the sender (i.e.,. the TEMPO server, sees the new values instantly). See the PCL function rdxGetTTLIN(). Only the low two bits of the argument n are significant. This function ignores all other bits. Valid values for n are 0, 1, 2 and 3. Since this function changes the TTL hardware at the time it is called, any changes to the TTL bits take effect immediately.

dx_GetTTLIN - Reads the I/O port and returns the two bits associated with the RDX TTL inputs. These are two TTLs that the sender can set and the receiver can read. In your TEMPO protocol, see the rdxSetTTLOUT() PCL function to set the bits. The return value of dx_GetTTLIN() is 0, 1, 2 or 3.

Notes About 8255 Digital I/O Chip

The 8255 MODE 0 is used for both sending and receiving 8255's. The RTS/CTS style protocol for sending 16 data bits in one direction is used.

Either end of a connection may be started first; the protocol is designed to not lose the first byte/word when a connection is established regardless of which is started first. Either sender or receiver can call dx_close at any time.

A simple, straight through, cable connecting the 24 digital lines + ground is all that is necessary.

Do not connect two RDX senders to the same cable at the same time.

> DON'T CONNECT +5,-5,+12,-12, etc. BETWEEN COMPUTERS

Error Detection and Response

Both sender and receiver are specifically designed to allow either to be started first without losing the first byte. Of course, with only one end started, the signal on the other floats (can vary randomly) so it is entirely possible that a correct sequence of changes gives the appearance to the end that is running that the other is also running even when it isn't. This is unavoidable and is also very unlikely to happen. The probability of seeing a correct status sequence from a dangling cable is reduced as the correct sequence for data transfer is made more complex (longer).

It is up to you to implement error detection, correction, flow control, etc. functions of any higher level protocols. These routines provide a low level 8 bit data path over which you may receive your data. Data being transmitted is not removed from the transmit buffer until it is received and stored in the receive buffer.

Waiting for Unlatched Values

Since RDX does not use hardware latching (i.e., mode 0 is not latched either on input or output), when a program is waiting for input, it reads the status several times until it doesn't change (i.e., until it is stable). This avoids catching the status inside the (roughly) 100ns transition period while the sender toggles the lines to their new value. Believe it or not, we *must* do this because we were, in fact, catching the status during its transition!

Interrupts

Processor interrupts are disabled (only if they are not already disabled) and reenabled briefly. This is done only when a character is enqueued or dequeued from a buffer. This allows these routines to be used in an interrupt service routine.

Testing

The DDX.EXE utility program may be used to transmit a file of data to an RDX receiver. Alternatively, you can use TEMPO's dsend() and dsendf() PCL functions to receive characters from a TEMPO protocol.

Running RDX Library Applications on Windows NT, 2000 and XP

DOS and Windows 98 allow direct access to I/O ports so we recommend that you use Windows 98 if you want to use RDX with Windows. To use RDX with Windows, you must install and start the GIVEIO.SYS device driver. You will find the required files on your RDX Library distribution media.

The RDX Library is not supported by Reflective Computing on the Windows NT, 2000 or XP platforms because these operating systems do not allow direct access to the I/O ports of the digital i/o cards.

If you want to try to use RDX on any of these operating systems, you must install the GIVEIO.SYS driver. The GIVEIO.SYS driver enables direct port access to your application program on these Windows systems. The GIVEIO.SYS (and related utilities such as LOADDRV.EXE) are *not* supported by Reflective Computing.

NOTE
These instructions apply only to using the RDX library on Windows NT, 2000 and XP.
Please do *not* following these instructions if you are running
your application on a DOS, Windows 95 or Windows 98 computer.

To install the GIVEIO.SYS device driver, follow these steps.

1. Copy GIVEIO.SYS (from your RDX distribution media) to your Windows (e.g., C:\WINDOWS) directory.
2. Run LOADDRV.EXE (from on your RDX distribution media)
3. Enter the FULL path name to the GIVEIO.SYS driver. For example, C:\WINDOWS\GIVEIO.SYS
4. Select INSTALL. You should see "successful".

If you are running Windows NT, 2000 or XP, you must tell Windows to start the GIVEIO driver each time you boot the computer. However, the following procedure can be used to tell Windows to start the GIVEIO driver at boot time (at the time the computer is started).

1. Right Click MyComputer icon
2. Select Properties/Hardware
3. Click DeviceManager button
4. Select View/ShowHiddenDevices
5. Open Non-Plug and Play Drivers
6. Double click GIVEIO
7. Click Driver tab
8. Under StartType, select BOOT
9. Click OK, OK and dismiss Device Manager

Reboot computer and Windows should automatically start the GIVEIO driver. The device driver is now permanently installed into Windows and should start each time you start your computer. This procedure only needs to be done once.

Manually Starting GIVEIO. Before running your program that uses RDX (or the RDX test program RDXTST32.EXE) on Windows, the GIVEIO device driver must be started. You can do this in any of several ways.

1. In the ControlPanel/Devices dialog, you can also tell Windows to start GIVEIO each time the system is booted so you don't have to manually do this (described above).
2. NET START GIVEIO. If you start GIVEIO this way, you will need to execute this command each time you boot your Windows XP computer.
3. Use ControlPanel/Devices. Locate the GIVEIO driver and START it.

Once GIVEIO is started, you can run RDXTST32.EXE or your Win32 RDX application.

Windows NT, 2000 and XP with Instacal 5.13:

Running Instacal version 5.13 disables the GIVEIO driver and prevents RDX from accessing the i/o ports. To reenable GIVEIO, do the following after you have run Instacal:

1. Right Click MyComputer icon
2. Select Properties/Hardware
3. Click DeviceManager button
4. Open DAS Component. You should see PCI-DIO48H (or your PCI-DIO card)
5. Right click on the PCI-DIO card
6. Select Disable and click YES when Windows asks if you really want to disable the PCI-DIO card. You should now see a red X to the left of the PCI-DIO card
7. Right click on the PCI-DIO card again
8. Select Enable
9. Close device manager and click OK in SystemProperties dialog

Your RDX application should now run.

Obtaining Base Address of your PCI Card

This section describes how to obtain the base address of the PCI card from a C program in Windows NT, 2000 and XP.

The RDX library requires the base address of the PCI card. Within your C program, you must use Computer Board's Universal Library to get the base address of the PCI card. The base address is obtained by calling the cbGetConfig() function to determine the base address of the PCI card. Set DevNum = 1 to get the base address space of the I/O range assigned to the board.

See the CBSHOW.C program on your RDX distribution. CBSHOW provides an example on how to obtain the base address of the 8255 chips.

CBSHOW.EXE is a convenient utility that displays base address information for each card in your CB.CFG (Computer Boards/Instacal configuration file). For example, the output of CBSHOW on a Windows computer with a PCI-DIO48H board looks like:

```
CBSHOW 1.2 - Copyright 2001 Reflective Computing.  All Rights Reserved.
Getting information for all Computer Boards cards..
```

```
CB Board #0 : IRQ 0    PCI-DIO48H
  PCI BASE ADDR 1-6:  EC80 EC60 0000 0000 0000 0000
  8255 (DIO) Base Address: 0xEC60 0xEC64

1 boards detected
```

The base address of the first 8255 chip is, in this example, 0xEC60. The base address of the second 8255 chip is, in this example, 0xEC64.

If CBSHOW does not recognize your DIO card, run Instacal and make sure that the card is configured in the CB.CFG (Computer Boards configuration) file.

See note on "Windows NT, 2000 and XP with Instacal 5.13".

Base Address for RDXTST32

This section describes how to obtain the base address of the PCI card in Windows XP for testing the RDX connection with the RDXTST32.EXE test program without programming in C.

One method to obtain the base address of a Computer Boards PCI card is to use INSCAL32.EXE, Computer Board's Instacal program. Run INSCAL32 and right click the desired PCI-DIO card. Then select Configure... from the menu. The Base Address is listed in the Board Configuration dialog.

You can also obtain the base address of the PCI card from the Windows Control Panel's *System* applet. Please note that every time the computer reboots, the PCI bus assigns the base address to the PCI card. So the base address can change when you reboot the computer. This is the reason why you should not embed the base address you get from this procedure in your program: It may change the next time you reboot the computer!

1. Click Start/Settings/ControlPanel/System
2. Select Hardware tab
3. Click Device Manager Button
4. Highlight the PCI DIO card that you want to use
5. Right click and select Properties
6. Select the Resources tab
7. The base address is the first hex number in the Input/Output range. For example, you might see the range as ECC0-ECFF. The base address is EC00.

You can now run RDXTST32 with the base address of 0xECC0:

```
rdxtst32 0xECC0
```

T8255 and T8255NT Test Programs

The T8255.EXE and T8255NT.EXE programs are now available (upon request) for testing the RDX link. T8255.EXE must be used with DOS or Windows 98 and T8255NT.EXE must be used with Windows NT, 2000 and XP.

If you are having problems getting RDXTST or RDXTST32 to work, you can test the RDX link using T8255 and T8255NT. T8255/T8255NT are low level diagnostic programs for boards with 8255 chips.

1. On the server, boot to DOS mode and type at the prompt:

```
T8255 i pci-dio48h/ctr15,0,0
```

T8255 in "input" mode watches the 8255 chip and displays the TTL values. Whenever the values change, T8255 displays them on a new line.

2. On the RDX computer, obtain the base address of the first 8255 chip on your PCI-DIO card with CBSHOW (in the example above, chip 0 is 0xEC60).

3. If the RDX computer is running Windows 98, run T8255 in a DOS window with:

```
T8255 o 0xEC60
```

If the RDX computer is running Windows XP, run T8255NT in a DOS window with:

```
T8255NT o 0xEC60
```

When T8255 (or T8255NT) is run in "output" mode, it toggles the TTLs on the 8255 chip once per second.

If GIVEIO is installed and working properly, you should see the TTLs change on the server computer as well as on the RDX computer.

If this test fails, the problem could be any of the following:

1. Incorrect base address on RDX computer or on server
2. Faulty DIO card
3. Faulty cable
4. Faulty cable connection
5. The GIVEIO driver is not installed/started on RDX computer (Windows XP)

An easy hardware test that bypasses Windows XP and GIVEIO driver. This test will help you to identify whether the problem is with your hardware (computer, base addresses, DIO board, cable) or if it is a software driver problem (GIVEIO not installed or started or you ran Instacal without disabling and reenabling the PCI-DIO board (see above).

1. Create a DOS boot diskette on a Windows 98 computer: In a DOS window on a Windows 98 computer:

```
format a:/s
```

Copy T8255.EXE and DDX.EXE from your TEMPO Utilities diskette onto the newly formatted diskette.

2. Reboot your RDX (Windows XP) computer with the DOS boot disk.

3. Type at the DOS prompt:

```
A:\T8255 O PCI-DIO48H,0,0
```

If you see the TTL bits toggling on the server computer, then all the hardware is working and the base addresses are correct. The problem is with the GIVEIO installation. See notes above on how to install and start GIVEIO.