

---

---

***UART4***

---

---

---

---

---

---

## **4-Port Serial Interface for the Persistor CF-1**

**Oceanographic Embedded Systems**

**1260 NE Seavy Ave.  
Corvallis, OR 97330**

**e-mail: [mark@oes.to](mailto:mark@oes.to)**

**Revision 1.1 October, 2001**

## Introduction

The UART4 is a peripheral interface for the Persistor CF-1 that provides four channels serial I/O with user-selectable baud rates and either polled or interrupt-driven data transfer. The interface card also provides a 3V lithium coin cell to provide backup power for the CF-1 real-time clock and a 10-pin ribbon cable connector for serial I/O.

## Setup and Testing

Setting up the UART4 is as simple as plugging the CF-1 into the PCB, connecting power, CF-1 serial I/O user serial inputs, then running one of the test programs.

### CF-1 Serial Port Connections

The serial I/O connector provides the same RS-232 signals implemented on the Recipe cards from Persistor. The header is compatible with the IDC-10 to DB9F connector also provided by Persistor. The pin connections of this cable are shown in figure 1.

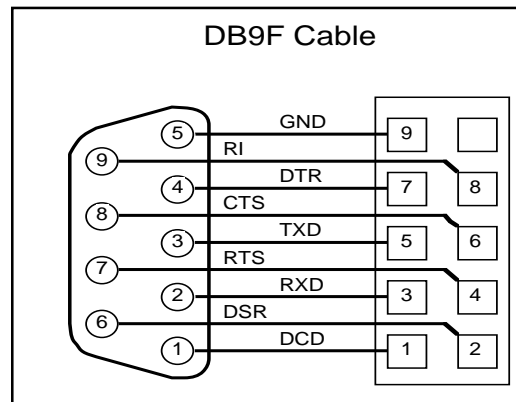


Figure 1. IDC10 to DB9F Cable

### Driver Expansion Connections

Your UART4 is shipped with shunts on the Driver Expansion Connector to enable and utilize the onboard MAX3243 RS-232 transceivers. The expansion connector also provides other signals—both from the UART and the CF-1—which can be used to control serial port handshaking or RS-485 drivers.

### Power Supply Input

Power is provided to the UART4 and the CF-1 through a 2-pin screw-terminal connector, J1. Although the CF-1 regulator will block reversed voltages at its inputs, you are likely to damage any other components connected to the supply.

NOTE: The +Voltage input is nearest the center of the board. The ground lead is nearest the edge of the board.

When operating at full speed and writing to the compact flash disk, the CF-1 can draw up to 70mA. The UART4 adds only about 20mA—depending on the loading of the RS232 ports connected to the card. You should use a current-limited supply set to about 100mA when developing and testing a new system.

The UART4 and CF-1 can accept input voltages to 20Volts without damage. However, if you have a power supply of greater than 11 Volts, it would probably be best to implement a step-down switching converter to provide about 7 volts to the CF-1 and UART4. A reasonably efficient converter will reduce the current drain on the supply—which can greatly extend the life of a battery supply. Please contact OES if you have a supply of greater than 12Volts where current drain is important.

## Test Software

The diskette provided with the UART4 includes a number of compiled applications that you may use to test your UART4 and CF-1. You may also download this software, and other example programs as they are developed, from the OES web page ([www.oes.to](http://www.oes.to))

## Hardware

The UART4 is assembled on two-sided printed circuit board using both surface-mount and through-hole components. The arrangement on the components on the board is shown in figure 2.

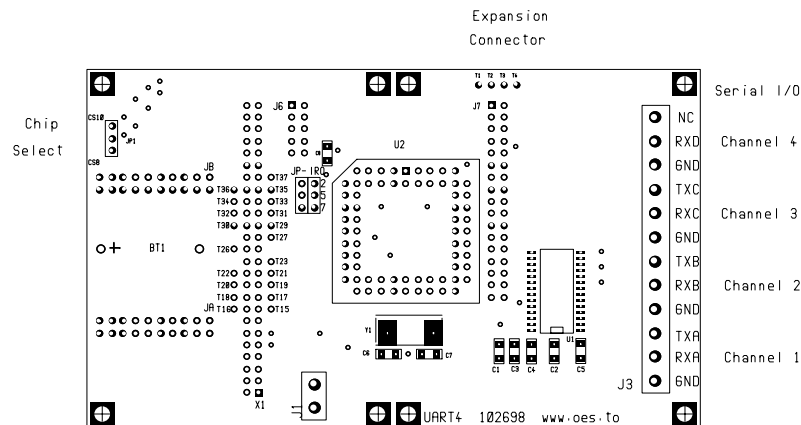


Figure 2. Component layout on UART4 PC board.

## Startech 16C554D Quad UART

The Startech 16C654 is a quad UART integrated circuit that is designed to interface both to IBM PC and Motorola 68000 buses. The IC provides each channel with a separate baud rate generator, control and configuration registers and 64-byte First-In-First-Out (FIFO) buffers. These FIFO registers greatly reduce the number of interrupts that the processor must handle during high-speed communications sessions.

The 16C654 is powered directly by the +3.3V supply from the CF-1. It normally requires only about 3mA during operation. Since there is no way to shut off power to the UART during operation, you cannot eliminate the power drain of the UART except by putting the CF-1 into the suspend mode—which turns off the +3.3V supply. It is not possible to simply turn off power to the 16C554, since it is directly connected to the address and data busses of the CF-1's microcontroller.

## UART4 Serial I/O Connections

The UART4 provides four sets of screw terminal connectors for Serial I/O. Pin 1 of each set is connected to the system ground. Pin 2 is the receive input and Pin 3 is the transmit output from the UART4. Note that channel 4 does not provide a transmit output, due to the limitations of the single driver IC used on the UART4.

## Driver Expansion Connector

If you require something other than the RS-232 interface provided, you may implement a mezzanine plug-in card to connect to the 34-pin header strip that provides the circuitry required for your application. The header also provides +3.3V power, a shutdown signal and several other outputs from the 16C554. These outputs may be used to control handshaking or RS-485 drivers. If no expansion card is used, the serial IO signals are connected to the onboard RS-232 driver with shunt connectors. Four pins are also connected to test pads which you may connect to CF-1 I/O pins or other signals. Figure 3. shows the pin assignments for the expansion connector.

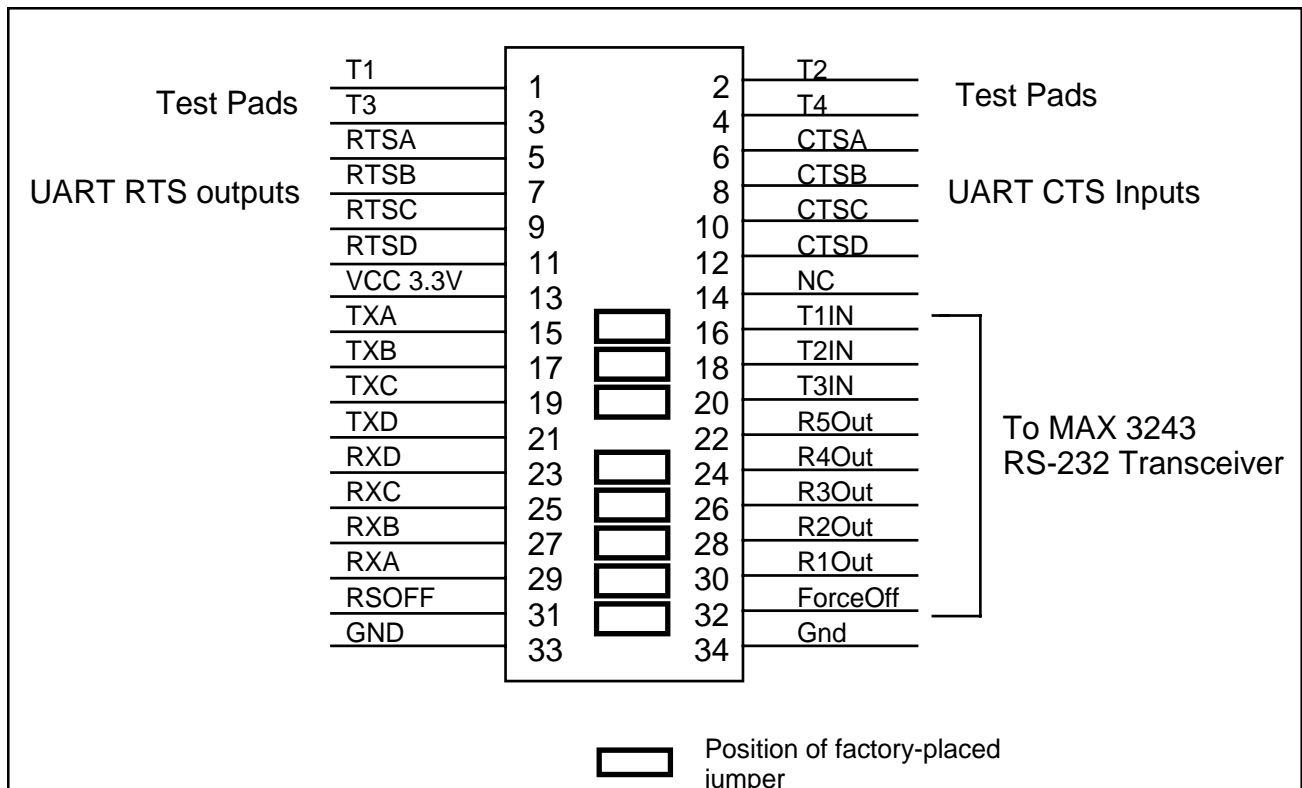


Figure 3. Expansion Connector Pin Assignment

## CF-1 Serial I/O Connector

The serial port header has the same pinout as the header on the PicoLog Recipe Card®. It is compatible with the ribbon-cable and 9-pin connector provided with that interface.

## Backup Battery

The UART4 provides a 2032 Lithium coin cell backup battery for the CF-1. If you are stacking a Recipe Card® or other interface along with the CF-1, only one of the cards should provide a backup battery. Otherwise a dead or defective battery on one card could drain a battery on another card.

## Chip-Select Jumper

The 16C654D UART is connected directly to the address and data busses of the CF-1. In order for it to operate properly, the UART must respond only to a unique set of addresses. The addresses selected for the UART are determined by the programming of either the CS8 or CS10 chip-select outputs of the 68338 MCU. The UART4 is normally shipped with the jumper at JP1 connected to utilize CS8.

**WARNING:** You must have a jumper in place for either CS8 or CS10! If you remove the jumper and turn on the CF-1, the UART's chip select input will be left floating and will cause erratic operation of the CF-1.

## Interrupt Selection

When used in interrupt-driven operation, the UART4 can interrupt the processor using IRQ2, IRQ5 or IRQ7. The UART4 is normally shipped with IRQ2 selected. If you are using other interrupts in your system, you may select IRQ5 or IRQ7 by moving the jumper. IRQ7 is the highest priority interrupt.

## Stacking CF-1 Connector Option

The UART4 can optionally be provided with stacking connectors for the CF-1. These connectors allow the CF-1 to be used with other Recipe Cards® or interfaces from OES. If you decide to stack multiple cards on the CF-1 it becomes your responsibility to ensure that there are no conflicts in the pin usage on the CF-1.

# Software

## UART4 Library

A library of software routines for the UART4 is provided in C-Language source code format for the MetroWerks CodeWarrior development system. The library files (u4.c and u4.h) provide routines to initialize the quad UART and to send and receive data from the serial I/O ports. The serial ports may be operated in either polled mode or in interrupt-driven mode with buffer sizes you select.

## Library Functions

The following functions are defined in the U4 library. The header file for the library, U4.h, is included in this document as appendix B.

---

### Function Prototype

```
void U4InitCS8(ulong baseaddr);
```

```
void U4InitCS10(ulong baseaddr);
```

### Parameters

*baseaddr* sets the address at which the hardware registers of the 16C654 UART will be located. Since the registers occupy only 64 bytes, the chip select will be assigned its minimum block size of 2k bytes. There are two memory areas available in the CF-1 memory map: a 14MB area at 0x00100000, and a 24KB area at 0xFFFF8000. It seems sensible to use the upper area for the UART4. If you have a second memory-mapped peripheral, you will have to ensure that the selected memory areas do not overlap.

### Usage

This function initializes the designated chip select registers for the UART4. Since the 16C654 is fast enough, no wait states are used. The chip select is set up for an 8-bit peripheral with read and write access and synchronized to AS (Address Strobe). The function also initializes a private variable with the base address for use by the other functions. You should be sure that you call the function that matches the chip select corresponding to the setting of the CS jumper on the UART4.

---

### Function Prototype

```
void U4SetInterrupt(short irq);
```

### Parameters

*irq* selects which hardware interrupt will be used. You must select the interrupt (from 2, 5 or 7) which matches the position of the hardware jumper on the UART4.

### Usage

This function inserts the address of the interrupt handler into the CF-1 exception vector table and sets the proper IRQ pin to function as an interrupt input. The function should be called AFTER the chip select is set up and any input or output queues are initialized.

---

## Function Prototype

```
bool U4InitInputQueue(short chan, short qsize);
```

### Parameters

*chan* selects which of the UART channels to initialize (1 through 4).

*qsize* sets the size of the queue to be used with the channel. Reasonable values should be in the range from 128 to 4095. Memory for the data in the input queue is allocated in the system heap with a call to `malloc()`.

### Return Value

The boolean result is true if the queue is successfully initialized. It is false if the memory for the queue data could not be allocated.

### Usage

This function is used to define queue sizes for an input channel and to set up the channel for interrupt-driven I/O. If you do not call this function, the channel will operate in polled mode.

---

## Function Prototype

```
bool U4InitOutputQueue(short chan, short qsize);
```

### Parameters

*chan* selects which of the UART channels to initialize (1 through 4).

*qsize* sets the size of the queue to be used with the channel. Reasonable values should be in the range from 128 to 4095. Memory for the data in the input queue is allocated in the system heap with a call to `malloc()`.

### Return Value

The boolean result is true if the queue is successfully initialized. It is false if the memory for the queue data could not be allocated.

### Usage

This function is used to define queue sizes for an input channel and to set up the channel for interrupt-driven I/O. If you do not call this function, the channel will operate in polled mode.

---

## Function Prototype

```
void U4Configure(short chan, long baud, char parity,char  
bits, char stop);
```

## Parameters

*chan* selects which of the UART channels to initialize (1 through 4).

*baud* sets the baud rate. You can select standard baud rates from 300 to 230,400 baud. Both the input and output channels will use the selected baud rate.

*parity* defines the parity of the characters transmitted and received. Valid settings are :

- 'n' no parity bit used
- 'e' set parity bit so an even number of logic 1s are transmitted
- 'o' set parity bit so an odd number of logic 1s are transmitted
- '0' set parity bit to 0
- '1' set parity bit to 1

*bits* sets the number of bits in the character. Only values of 5 through 8 are valid. Only values of 7 and 8 are used in normal computer-to-computer applications.

*stop* sets the number of stop bits. Only values of 1 and 2 are valid.

## Usage

This function must be called before the channel is used, to properly define the baud rate and character format.

---

## Function Prototype

```
void U4TxPutChar(short chan, ushort data);
```

## Parameters

*chan* selects which of the UART channels that will transmit the data(1 through 4).

*data* specifies the character to send. The parameter is of type ushort because the standard libraries use and INT or SHORT—but you will usually be transmitting characters.

## Usage

This function is used to transmit data through the UART channel. In polled mode, it will wait inside the routine until the UART transmit register is empty (up to one character time). In interrupt-driven mode, the character will be put into the queue to be transmitted by the interrupt handler.

---

## Function Prototype

```
ushort U4RxGetChar(short chan);
```

### Parameters

*chan* selects which of the UART channels will receive the data (1 through 4).

### Return Value

The return value is the character received (promoted to a ushort).

### Usage

In polled mode, the function will wait until a character is received. This could hang your program! In interrupt-driven mode, the routine will return the character 0x00 if there is no received data available. For this reason, you should always call U4RxCharsAvail() to check for received data before you call U4RxGetChar().

---

## Function Prototype

```
ushort U4RxCharsAvail(short chan);
```

### Parameters

*chan* selects which of the UART channels to check for incoming data(1 through 4).

### Return Value

The return value is the number of receive characters available.

### Usage

In polled mode, the return value will be one when there is a received character available. Since the 16C654 has a 64-byte receive FIFO, you may have to call U4RxCharsAvail() and U4RxGetChar() many times to get all the characters received. In interrupt-driven mode, the function returns the number of characters in the receive queue. There may be characters in the UART FIFO which have not yet been added to the input queue.

---

## Function Prototype

```
void U4Close(void);
```

### Parameters

NONE

### Usage

This function disables the interrupts—both at the IRQ pin and in the interrupt-enable registers of the UART. It also frees the memory (if any) allocated for input or output

queueus.

---

### Function Prototype

```
short U4XmitErrorCode(short chan);
```

```
short U4RcvErrorCode(short chan);
```

### Parameters

*chan* selects which of the UART channels will report errors (1 through 4).

### Return Value

The return value is an indication of the most recent error..

### Usage

This function returns a unique code for each error. The code is cleared to 0 when the error code is read. 0 = NoError 1 = tried to put data in full queue 2 = tried to get data from empty queue 4 = illegal channel number.

---

### Function Prototype

```
void U4RTSState(short chan, short state);
```

### Parameters

*chan* selects which of the UART channels will change its clock divisor.

*state* determines whether the DTR or RTS bit is a one or a zero.

### Usage

These functions are used to set and clear the RTS bits of the UART channels. The RTS output is often used for input handshaking or RS-485 output driver enable. Input handshaking is seldom required with the CF-1 and UART4 due to the fast interrupt response of the CF-1 and the 64-byte receive FIFO of the 16C654.

---

## Function Prototype

```
short U4CTSSState(short chan);
```

## Parameters

*chan* selects which of the UART channels will report CTS level(1 through 4).

## Return Value

The return value indicates the state of the CTS input of the channel (0 or 1).

## Usage

This function returns the level of the CTS input. You may be able to use this input to set up transmit handshaking, however such handshaking is complicated by the 64-byte transmit FIFO of the 16C654.

## Sample Programs

The diskette provided with UART4 provides source and compiled .RUN code for several sample programs illustrating the use of the UART4. These must be loaded into the CF-1 using the MotoCross loader from Persistor.

### U4File

This program uses Serial Channel 1 to transmit data at 19200 baud and receives the same data through channels 1 through 4. The input data from each channel is stored in a separate file on the compact flash disk attached to the CF-1. In order to collect data, you must connect jumper wires from the TX output of channel 1 to each of the four RX inputs.

