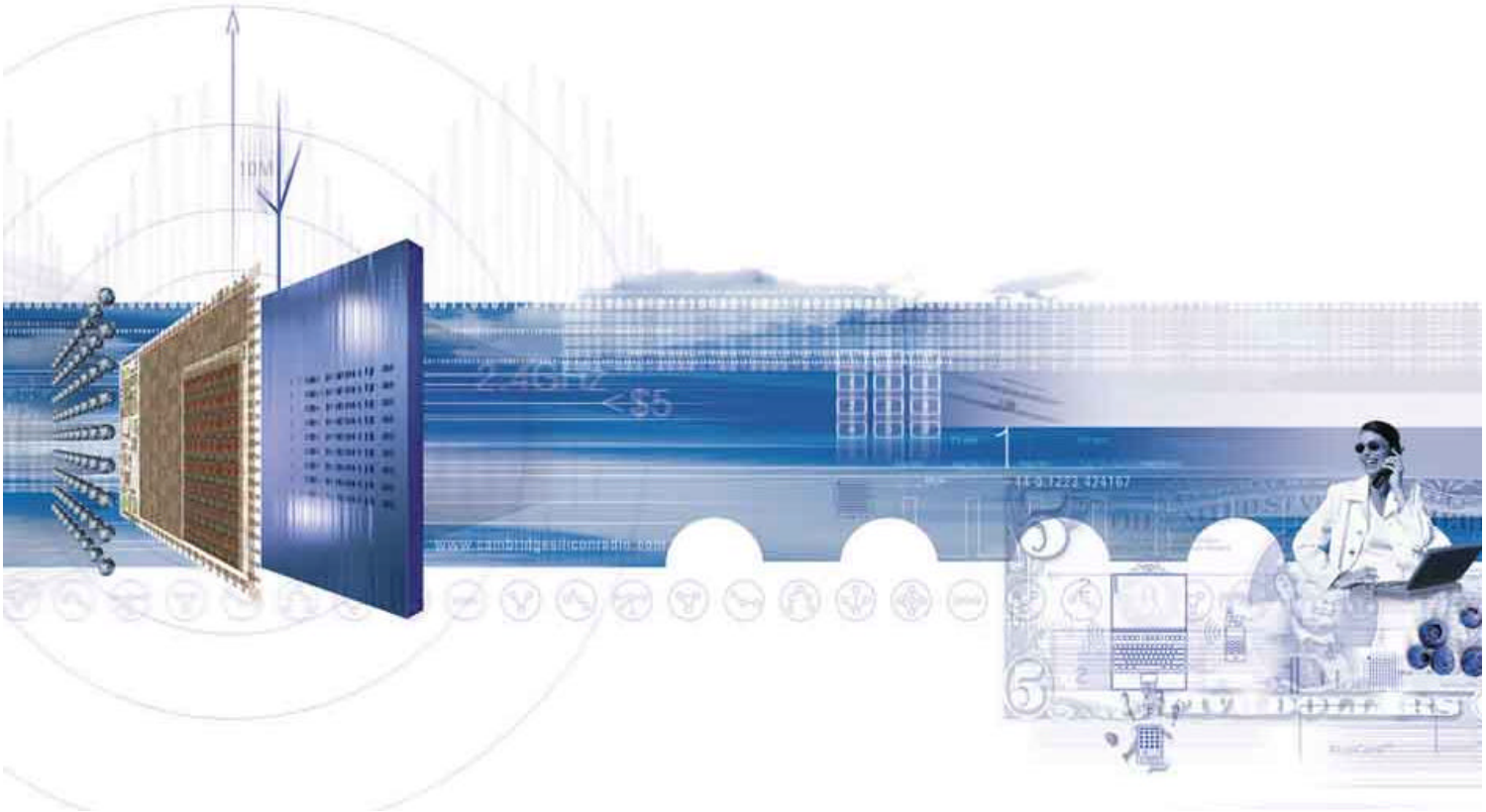
BlueCore ä 01

UART, PCM and PIO Interfacing Application Note

AN019

August 2001



CSR

Unit 300 Cambridge Science Park
Milton Road
Cambridge
CB4 0XL
United Kingdom

Registered in England 3665875

Tel: +44 (0)1223 424167
Fax: +44 (0)1223 424178

www.csr.com

Bluetooth Qualified

Purpose

As a supplement to the **BlueCore™01** datasheet. This application note provides further details on **BlueCore01** interfaces commonly used by applications engineers. Appendix A provides an application circuits intended to help decrease the design time of **BlueCore01** applications.

Off-Chip Program Memory Interface

The external memory port provides a facility to interface up to 4Mbits of 16-bit external memory. This off-chip storage is typically used to store **BlueCore01** settings and program code. External RAM/ROM is supported by this interface with FLASH being the storage mechanism typically used by **BlueCore01** modules.

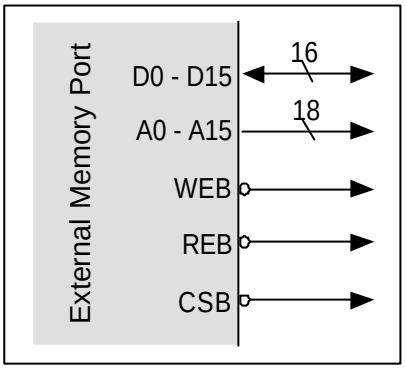


Figure 1: Extended Memory Interface

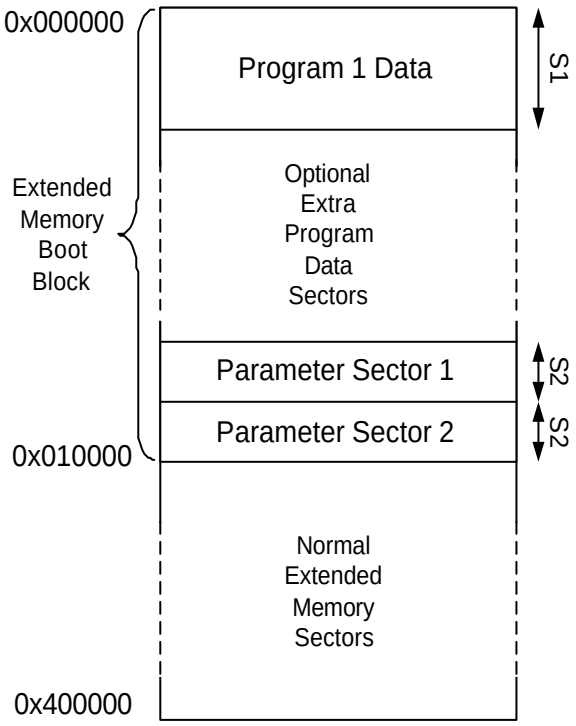


Figure 2: Typical Extended Memory Organisation

The external memory port consists of 16 bi-directional data lines, 18 output address lines and three active low output control signals, as shown in Figure 1. The function of the control signals is quite simple. *WEB* is asserted when data is written to external memory; *REB* is asserted when data is read from external memory and the chip select line, *CSB*, is asserted when any data transfer (read or write) is required. All of the external memory port connections are implemented using CMOS technology and use standard 0V and VDD (+3.0V nominal) signaling levels.

When choosing an external memory device that will interface to **BlueCore01**, the selected chip should meet the following guidelines.

Property	*Value*
Data Word	16-bit
Minimum Total capacity	4Mbit (256kB)
Maximum Access Time	90ns

Table 1: External Memory Properties

In addition to these hardware requirements, particular care should be taken to ensure that the sector organisation of the extended memory has the correct format. A sector is defined as an individually erasable area of extended memory. **BlueCore01** assumes a boot block exists in the bottom 64k-word of extended memory that consists of 2 parameter sectors and at least one program data sector, as shown in Figure 2. The sizes of these sectors are detailed in Table 2.

Sector	Min	Max	Units
S1	16	56	k-word
S2	4	24	k-word

Table 2: Extended Memory Sector Sizes

BlueCore01 software drivers currently support SST, AMD, Fujitsu and Intel FLASH chips. When choosing memory chips to interface with **BlueCore01**, ensure that command interface is compatible with any of the supported chip types. Alternatively, **BlueCore01** drivers can be updated to support other devices if high volume orders are desired.

UART Interface

BlueCore01 Universal Asynchronous Receiver Transmitter (UART) interface provides a simple mechanism for communicating with other digital devices using the RS232 standard.

Four signals are used to implement the UART function, as shown in Figure 3. When **BlueCore01** is connected to another digital device, UART_RX and UART_TX transfer data between the two devices. The remaining two signals, UART_CTS and UART_RTS, can be used to implement RS232 hardware flow control, a standard method of preventing buffer overflows on RS232 devices. Overflow prevention is achieved by a simple handshaking mechanism, described by the following protocol:

1. When **BlueCore01** cannot receive any more data, it drives its UART_RTS (Ready To Send) output to 3V VDD.
2. **BlueCore01** only transmits data when the UART_CTS (Clear To Send) input is at 0V.

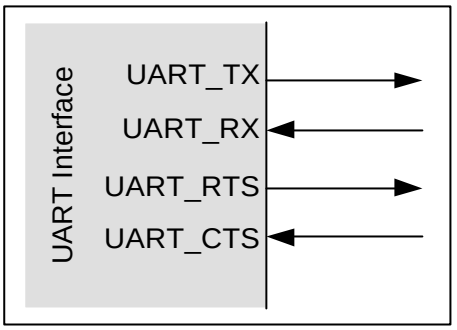


Figure 3: UART Interface

BlueCore01 is also shipped with firmware that implements software flow control. All UART connections are fabricated using CMOS technology and have signalling levels of 0V and VDD (+3V nominal).

With first shipments of the **BlueCore** evaluation system, the **BlueCore01** chip is pre-configured with a baud rate of 115.2kbps. Future upgrades of **BlueCore01** firmware will allow the UART baud rate and other configuration parameters to be selected from the PC user interface using a private communications logical channel running over the UART. Table 3 details the possible settings. It should be noted that to communicate with the UART at its maximum data rate using a standard PC, an accelerated serial port adapter card is required.

Property	***Possible Values***
Baud Rate	Min 9600 baud, Max 1.536Mbaud, (supporting all values in this range)
Flow Control	RTS/CTS or none
Parity	On or Off
Number of Stop Bits	1 or 2

Table 3: Possible UART Settings

The UART port also carries a number of logical channels: HCI data (both SCO and ACL), HCI commands and events, L2CAP API, RFCOMM API, SDD-B API and chip management. These channels are combined into a robust tunneling protocol where each channel has its own software flow control and cannot block the others.

Finally, the UART interface is capable of resetting the **BlueCore01** upon reception of a BREAK signal. A BREAK is identified by a continuous logic high on the UART_RX pin, as shown in Figure 4. If tBRK is longer than a predefined constant, set by **BlueCore01** software, a reset will occur. This feature allows a host to initialise the system to a known state.

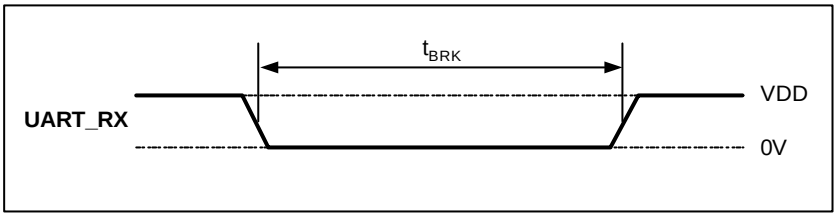


Figure 4: A BREAK Signal

Note: The reset-upon-BREAK feature is not enabled in the first shipments of the **BlueCore** evaluation system firmware.

PCM Interface

The PCM interface is used to transfer a single bi-directional voice channel between **BlueCore01** and a suitable CODEC chip or other hardware.

The PCM interface consists of four CMOS signals with signaling levels of 0V and VDD (+3V nominal), as shown in Figure 5. The voice samples are encoded as 2's complement 13-bit linear PCM at $8kss^{-1}$. The PCM_CLK and PCM_SYNC pins can be configured as either both outputs or both inputs, depending on whether the **BlueCore01** is the master or slave of the PCM interface.

When configured as the PCM master, **BlueCore01** generates an 8kHz PCM_SYNC signal and a 256kHz PCM_CLK signal. In this mode **BlueCore01** interfaces directly to the Motorola MC145483 CODEC chip.

When configured as the PCM slave, **BlueCore01** accepts an 8kHz PCM_SYNC signal and a PCM_CLK signal with a frequency between 128kHz and 512kHz.

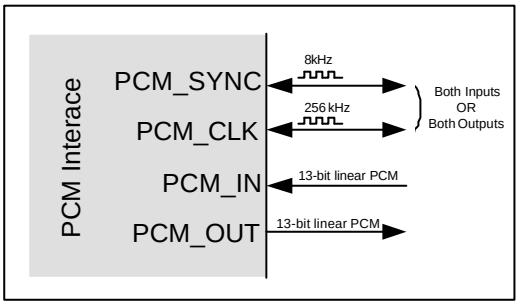


Figure 5: PCM Interface

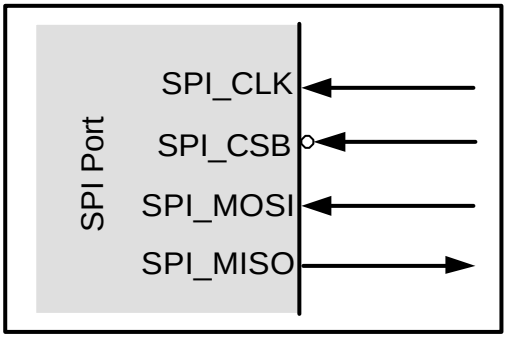


Figure 6: SPI Port

SPI Port

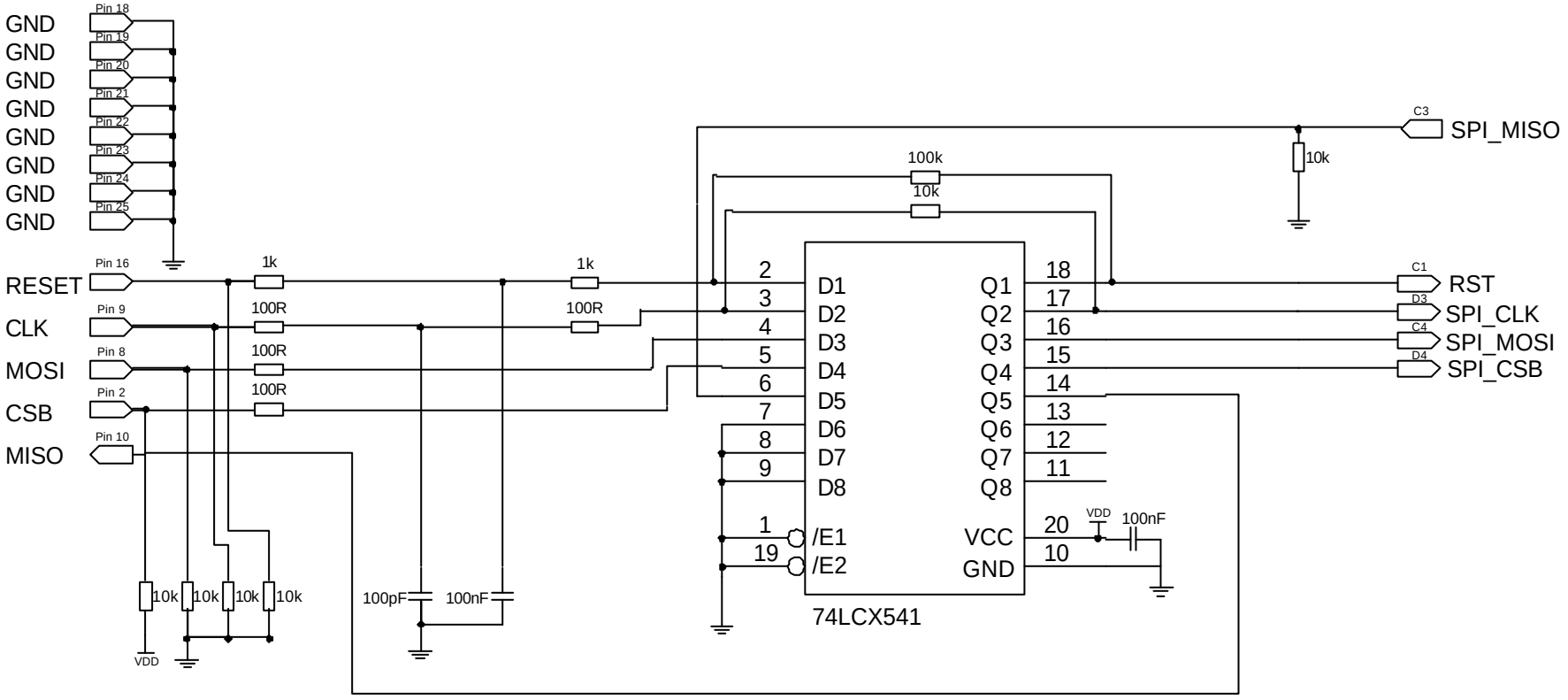
The **BlueCore01** serial peripheral interface (SPI) provides a mechanism for reading and writing internal registers, RAM and external memory. The SPI, communicating at up to 5Mbaud, is used to load, verify and debug **BlueCore01** software. The SPI port is the preferred method to load the **BlueCore01** firmware at the time **BlueCore01** is assembled onto the target circuit board. It is strongly recommended to route all SPI signals to contact points or a connector on the target circuit board. The SPI consists of four CMOS signals. This is consistent with the specification of the Motorola SPI standard. When **BlueCore01** communicates with an external host using the SPI, it always acts as the slave. Figure 6 shows that the SPI_CLK is generated by the host. SPI_MISO carries the data sent by **BlueCore01**. SPI_MOSI carries the data sent to **BlueCore01**. SPI_CSB is an active low input that must be asserted by the host whenever the SPI is used. Various PC software tools are available from CSR to download and debug development and application software using the SPI including a FLASH memory loader utility that is shipped with the **BlueCore** evaluation system. An application circuit is presented in Appendix A that illustrates how to connect the **BlueCore01** SPI to the parallel port of a standard PC.

PIO Port

The Parallel Input Output (PIO) Port is a general-purpose IO interface to **BlueCore01**. The port consists of eight programmable, bi-directional PIO[0:7] with a maximum current drive capability of 4mA. Two of these signals are typically used to enable or disable the radio transmitter and receiver. The PIO port can be used for a number of other applications such as:

- Support a user interface by connecting LEDs and switches to the PIO port
- Control external RF power amplifiers, RF low noise amplifiers and RF TX/RX switches.
- Toggle the PIO pins in accordance with the I^2C specification to achieve communication with an I^2C device.

Appendix A - SPI Application Circuit



Record of Changes

Date:	Revision:	Reason for Change:
20 APR 01	a	Original publication of this document (CSR reference: bc01-an-003a).
18 AUG 01	b	Corrected output and input voltages on page 6 (CSR reference: bc01-an-003b).

UART, PCM and PIO Interfacing

AN019

August 2001