CoreUART

DirectCore

Product Summary

Intended Use

- Basic Interface to Industry Standard UART Controllers
- Embedded Systems for Sharing Data between Devices with Limited Pin Counts Using Standard UART Protocols

Key Features

- Asynchronous (UART) Mode – Fully Programmable to any Baud Rate up to 1/16th of the System Clock Frequency with Glitch Rejection
- Synchronous Mode – 12 Clock Cycles Required per Byte Transfer
- 7 or 8 Bits of Data
- Parity (Odd, Even, None)
- Baud Rate Control for Asynchronous Mode
- Both Receive and Transmit are Double Buffered to Maximize Throughput

Targeted Devices

- MX Family
- SX Family
- SX-A Family
- eX Family
- RT54SX-S Family
- ProASIC/ProASIC^{PLUS} Family
- Axcelerator Family

Core Deliverables

- Netlist Version
  - Compiled RTL Simulation Model, Compliant with the Actel's Libero™ (IDE) Integrated Design Environment
  - Netlist Compatible with the Actel Designer Place-and-Route Tool (with and without I/O pads)
- RTL Version
  - VHDL or Verilog Core Source Code
  - Synthesis Scripts
- Actel-Developed Testbench (VHDL)

Synthesis and Simulation Support

- Synthesis: Exemplar, Synplicity, Design Compiler, FPGA Compiler, FPGA Express
- Simulation: Vital-Compliant VHDL Simulators and OVI-Compliant Verilog Simulators

Macro Verification

- Simulation Testbench

General Description

The CoreUART is a serial communication controller with a flexible serial data interface that is intended primarily for embedded systems. The controller can operate in either an asynchronous (UART) or synchronous mode. In the synchronous mode, the same UART protocols are used, but the baud rate is equivalent to the input clock frequency. When employing the CoreUART in the synchronous mode, the interacting devices must operate off of the same system clock. For the asynchronous mode, the clocks can be the same or different, including different frequencies. The main reason to use the synchronous mode is to improve data bandwidth.

In the asynchronous mode, the CoreUART can be used to directly interface to industry standard UARTs. The CoreUART is intentionally a subset of the full UART capabilities in order to make the function cost effective in a programmable device. Figure 1 on page 2 illustrates the various usages for the CoreUART.

Case A in Figure 1 on page 2 represents the interface to an industry standard UART like an 8251 or a 16550. For this case, the CoreUART must operate in an asynchronous mode and the baud rates of the UART must match a standard UART. Case B in Figure 1 on page 2 represents an embedded system. In case B, both CoreUARTs can operate either asynchronously or synchronously if CLKA = CLKB. If the clocks are different, then the UART must operate in asynchronous mode. Users need to ensure that the baud rates are equal for proper data transfers.

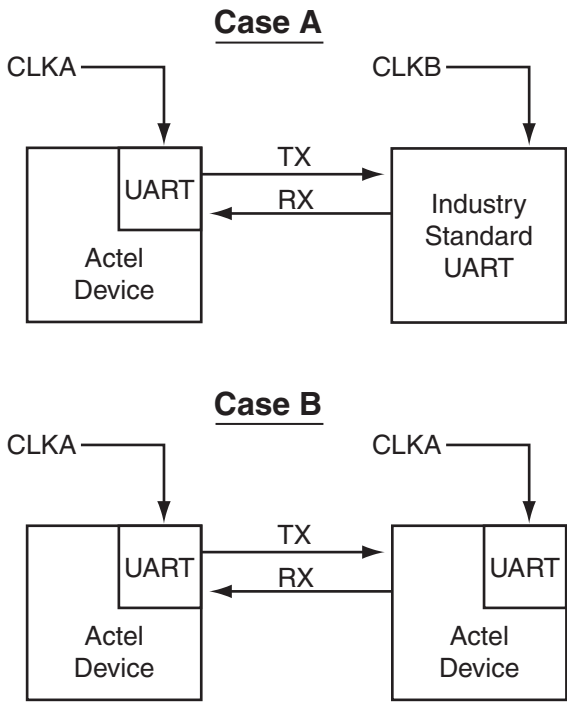


Figure 1 • *System Block Diagram Depicting CoreUART Usage*

Functional Block Diagram of CoreUART

Figure 2 shows the block diagram of the CoreUART core functionality. The baud generator creates a divided down clock enable that correctly paces the transmit and receive state machines. For the synchronous case, the baud generator is not utilized.

The function of the receive and transmit state machines are affected by the control inputs bit8, parity_en, and odd_n_even. These signals indicate to the state machines how many bits should be transmitted. In addition, the signals also suggest the type of parity, and if parity should be generated or checked. For asynchronous operation, the activity of the state machines is paced by the outputs of the baud generator.

To transmit data, the data is first loaded into the transmit data buffer. Data can be loaded into the buffer until the TXrdy signal is driven inactive. The transmit state machine will immediately begin to transmit data and will continue transmission until the data buffer is empty. The state machine first transmits a START bit, followed by the data (LSB first), then the parity (optional), and finally the STOP bit. The data buffer is double buffered, so there is no loading latency.

The receive state machine monitors the activity of the Rx signal. Once a START bit is detected, the receive state machine begins to store the data in the receive buffer until the transaction is complete, which in turn activates the receive_full signal, indicating valid data is available. Parity errors are reported on the parity_err signal (if enabled), and data overrun conditions are reported on the overflow signal.

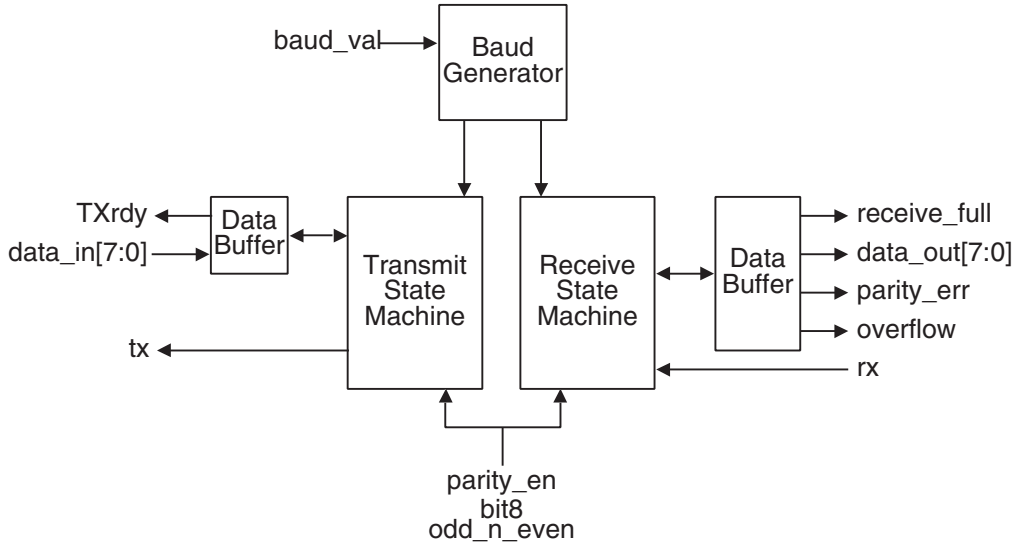


Figure 2 • *Block Diagram of the CoreUART Functionality*

I/O Signal Descriptions

Signal descriptions for the CoreUART are defined in Table 1. The signals are broken down into the following classes: system signals, parallel data transfer signals, serial control and status signals, and serial data signals. System signals consist of the CLK and reset_n signals. Parallel data transfer signals include data_in[7:0], data_out[7:0], WEn, OEn, and CSn. Control signals are bit8, parity_en, odd_n_even, and baud_val. Status signals are TXrdy, receive_full, parity_err, and overflow. The serial data signals consist of Rx and Tx.

Table 1 • *CoreUART Signals*

Name*	Type	Mode	Description
CLK	Input	Sync/Async	Main system clock
reset_n	Input	Sync/Async	Active low asynchronous reset
data_in[7:0]	Input	Sync/Async	Transmit write data bus
data_out[7:0]	Output	Sync/Async	Receive read data bus
WEn	Input	Sync/Async	Active low write enable. This signal indicates that the data presented on data_in[7:0] bus should be registered by the transmit buffer logic. This signal should only be active for a single clock cycle per transaction and should only be active when the TXrdy signal is active
OEn	Input	Sync/Async	Active low read enable. This signal is used to indicate that the data on data_out[7:0] has been read and will reset the receive_full bit and any error conditions (overflow or parity_err)
CSn	Input	Sync/Async	Active low chip select. The CSn signal qualifies both the WEn and OEn signals. For embedded applications, this signal should be tied to a logical '0'
bit8	Input	Sync/Async	Control bit for data bit width for both receive and transmit functions. When bit8 is a logical '1,' then the data width is eight bits; otherwise, the data width is seven bits and data defined by data_in[7] is ignored and data_out[7] is a don't care
parity_en	Input	Sync/Async	Control bit to enable parity for both receive and transmit functions. Parity is enabled when the bit is set to a logical '1'
odd_n_even	Input	Sync/Async	Control bit to define odd or even parity for both receive and transmit functions. When the parity_en control bit is set, a '1' on this bit indicates odd parity and '0' indicates even parity
baud_val	Input	Async	8-bit control bus used to define the baud rate
TXrdy	Output	Sync/Async	Status bit, when set to a logical '0,' indicating that the transmit data buffer is not available for additional transmit data
receive_full	Output	Sync/Async	Status bit, when set to a logical '1,' indicating that data is available in the receive data buffer to be read by the system logic. The data buffer controller must be notified of the reception by simultaneous activation of the OEn and CSn signals to prevent erroneous overflow conditions
parity_err	Output	Sync/Async	Status bit, when set to a logical '1,' indicating a parity error during a receive transaction. This bit is synchronously cleared by simultaneous activation of the OEn and CSn signals
overflow	Output	Sync/Async	Status bit, when set to a logical '1,' indicating a receive overflow has occurred. This bit is synchronously cleared by simultaneous activation of the OEn and csn signals
rx	Input	Sync/Async	Serial receive data
tx	Output	Sync/Async	Serial transmit data

Note: **Active low signals are designated with a trailing lower-case n.*

Device Utilization and Performance

Utilization statistics for targeted devices are listed in Table 2 and Table 3.

Table 2 • CoreUART Utilization in Asynchronous Mode

Family	Cells or Tiles		Utilization	
	Sequential	Combinatorial	Device	Total
Axcelerator	83	100	AX500	2%
SX-A	83	103	A54SX08A	24%
RT54SX-S	83	102	RT54SX32S	6%
ProASIC^{PLUS}	79	246	APA150	5%
SX	83	102	A54SX08	24%
ProASIC	79	246	A500K050	6%
42MX	79	86	A42MX09	24%
eX	80	97	eX128	46%

Note: *Data in this table achieved using typical synthesis and layout settings*

Table 3 • CoreUART Utilization in Synchronous Mode

Family	Cells or Tiles		Utilization	
	Sequential	Combinatorial	Device	Total
Axcelerator	59	65	AX500	2%
SX-A	59	67	A54SX08A	16%
RT54SX-S	59	68	RT54SX32S	4%
ProASIC^{PLUS}	57	143	APA150	3%
SX	59	68	A54SX08	17%
ProASIC	57	143	A500K050	4%
42MX	57	53	A42MX09	16%
eX	59	65	eX128	32%

Note: *Data in this table achieved using typical synthesis and layout settings*

CoreUART supports >75 MHz performance for all Actel FPGAs devices and >100 MHz for Actel's Axcelerator family devices.

Customization Options

RTL versions of the core can be customized for asynchronous and synchronous operations. For netlist versions, two versions of the core are provided: one for synchronous and the other for asynchronous operation.

Programmable Options

There are four programmable inputs in the CoreUART: baud_val (baud rate), bit8 (number of data bits), parity_en (parity enable), and odd_n_even (odd or even parity).

Number of Data Bits

The input bit8 is used to define the number of valid data bits in the serial bitstream. The most significant bit is a "don't care" for the seven bit case.

Parity

Parity is enabled/disabled with the input parity_en. When parity is enabled, then the odd_n_even input defines the type of parity.

Baud Rate

For the asynchronous mode, the baud rate must be specified. This is done by setting the value of the 8-bit baud_val bus. This value is a function of the system clock and the desired baud rate. The value should be set according to the following equation:

$$baudval = \frac{clk}{(baud \times 16)}$$

Where:

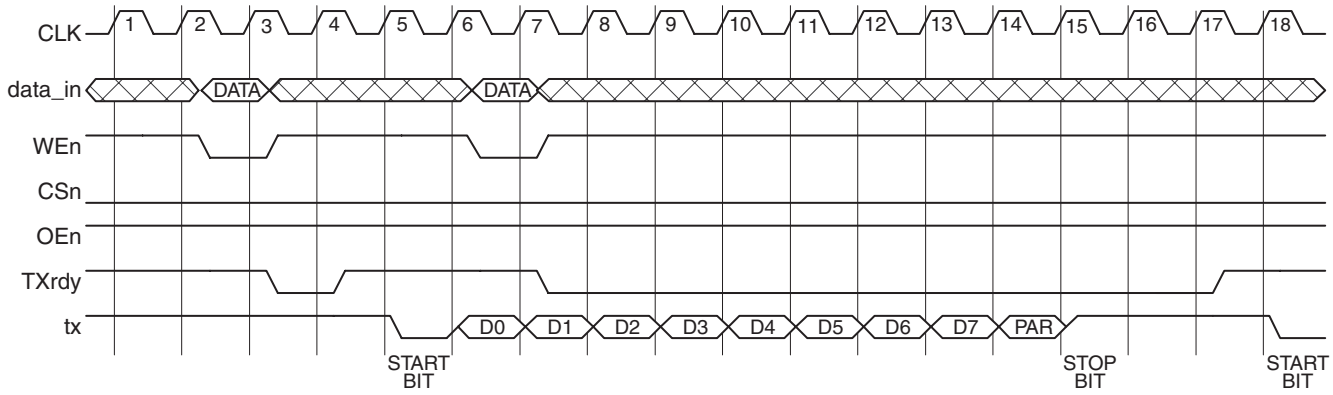
clk = the frequency of the system clock in hertz

baud = is the desired baud rate in hertz.

The term baudval needs to be rounded to the nearest integer. For example, a system with a 33 MHz system clock and a 9600 desired baud rate should have a baud_value of 215 decimal or D7 hex.

CoreUART Transaction

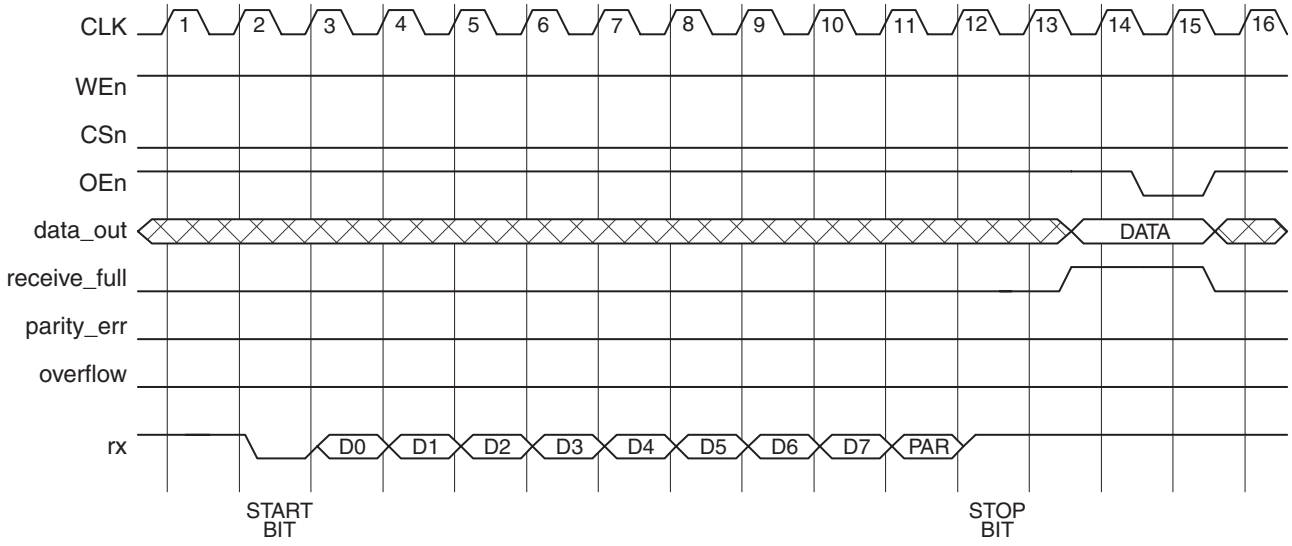
The UART's waveforms can be broken down into a few basic functions: transmit data, receive data, and errors. Figure 3 shows serial transmit signals, and Figure 4 on page 6 shows serial receive signals. Figure 5 on page 6 and Figure 6 on page 7 show the parity and overflow error cycles, respectively. To simplify the waveform description, all of the waveforms are shown in synchronous mode. Asynchronous transfers are similar; however, the serial transmission (START bit, data bits, parity bit, and STOP bit) require more than one clock cycle to complete. The number of clocks required is equal to the clock frequency divided by the baud rate. All waveforms assume that eight bits of data and parity are enabled.



Notes:

1. *A serial transmit is initiated by writing data into the CoreUART. This is accomplished by providing valid data and asserting the WEn and CSn signals. The TXrdy signal will become inactive for one cycle, while the data is being transferred from the transmit hold register to the transmit register that begins the serial transfer.*
2. *The transmission begins with a START bit, followed by data bits zero through six, the optional seventh bit, the optional parity bit, and finally the STOP bit.*
3. *Because the UART is double buffered, data can be queued in the transmit hold register (cycle 7). The TXrdy low line indicates that no more data can be transferred to the UART.*
4. *Once the previous serial transfer is complete, the data in the transmit hold register is passed to the transmit register and the transfer begins. The TXrdy line is also asserted indicating that the next data byte may be loaded.*

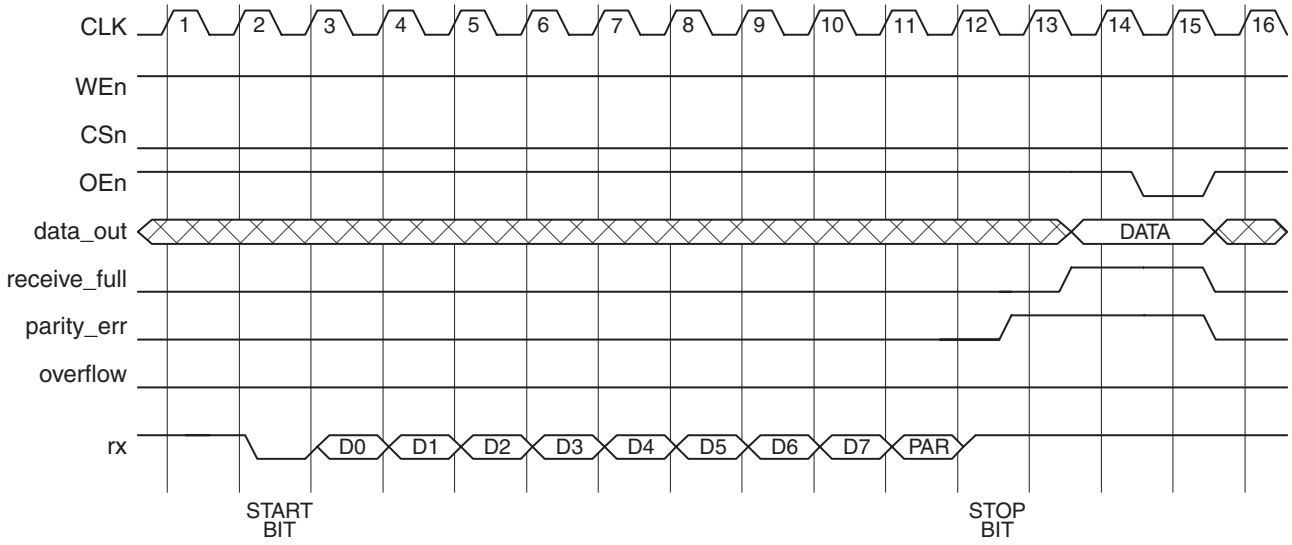
Figure 3 • *Serial Transmit*



Notes:

1. *The CoreUART continuously monitors the Rx line polling for a START bit. Once the START bit is detected, the CoreUART registers the data stream. The optional parity is also registered and checked.*
2. *Then the data is loaded into the receive hold buffer and the receive_full signal is asserted. The receive_full signal will remain asserted until the data is read externally, indicated by the simultaneous assertion of CSn and OEn.*

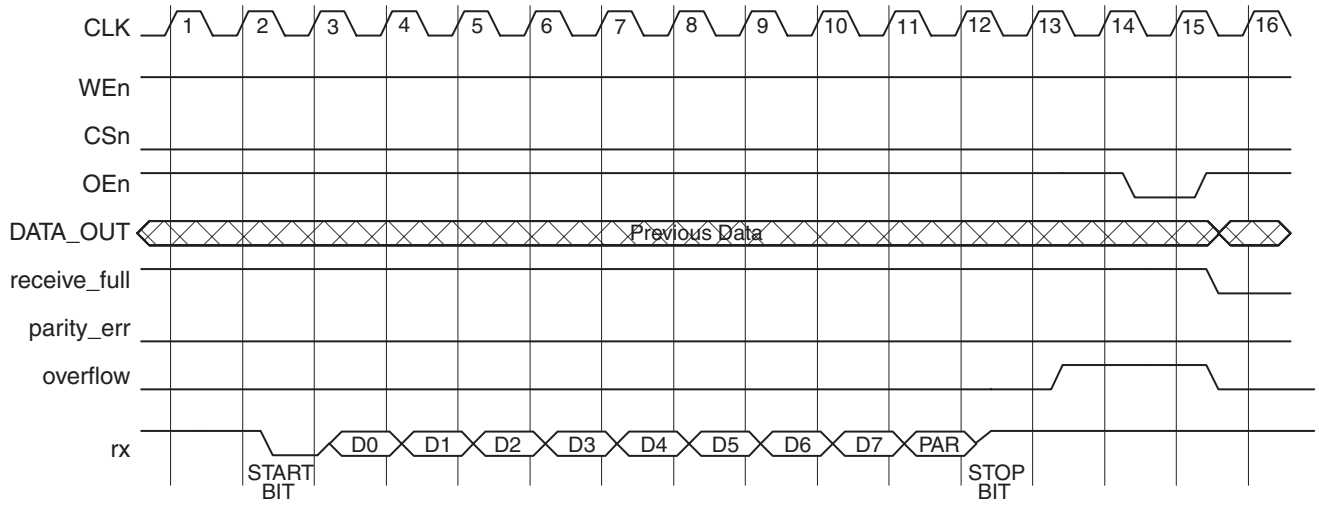
Figure 4 • *Serial Receive*



Notes:

1. *When a parity error occurs, the parity_err signal is asserted.*
2. *The error is cleared by the same method that data is read, simultaneous assertion of CSn and OEn.*

Figure 5 • *Parity Error*



Notes:

1. *When a data overflow error occurs, the overflow signal is asserted.*
2. *The previous data is held, and the new data is lost.*
3. *The error is cleared by the same method that data is read, simultaneous assertion of CSn and OEn.*

Figure 6 • *Overflow Error*

Ordering Information

Order CoreUART through your local Actel sales representative. Use the following numbering convention when ordering: CoreUART-XX, where XX is listed in Table 4.

Table 4 • *Ordering Codes*

XX	Description
EV	Evaluation Version
SN	Single-use Netlist for use on Actel devices
AN	Netlist for unlimited use on Actel devices
AR	RTL for unlimited use on Actel devices
UR	RTL for unlimited use and not restricted to Actel devices

List of Changes

The following table lists critical changes that were made in the current version of the document.

Previous version	Changes in current version (v5.1)	Page
v4.0	Table 1 on page 3 was updated.	page 3
	Table 2 and Table 3 on page 4 are new.	page 4

Datasheet Categories

Product Definition

This version of the datasheet is the definition of the product. A prototype may or may not be available. Data presented is subject to significant changes.

Advanced

This version of the datasheet provides nearly complete information for a prototype IP product. Code is fully operational, but may not support all features expected in the production release. A prototype core and a preliminary testbench are available.

Production (unmarked)

This version of the datasheet contains complete information on the final core. All components are fully operational and the core has been thoroughly verified.

http://www.actel.com

Actel Europe Ltd.
Maxfli Court, Riverside Way
Camberley, Surrey GU15 3YL
United Kingdom
Tel: +44 (0)1276 401450
Fax: +44 (0)1276 401490

Actel Corporation
955 East Arques Avenue
Sunnyvale, California 94086
USA
Tel: (408) 739-1010
Fax: (408) 739-1540

Actel Asia-Pacific
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Tel: +81-(0)3-3445-7671
Fax: +81-(0)3-3445-7668

5172143-3/12.02