Interfacing the Intersil X5163/323/643 CPU Supervisors to NEC 78K Microcontrollers

AN98
Rev 0.00
Jun 13, 2005

Introduction

The Intersil CPU Supervisors have an on-chip programmable watchdog timer and nonvolatile EEPROM memory. These features, coupled with the 3-line Serial Peripheral Interface (SPI) and the 78K-series micro-controller from NEC, make for an effective combination of features and performance. This application note will explore some possibilities and will provide example schematics and software.

Interface

The 78K-Series microcontroller typically has two serial ports. One of these, a synchronous three-line interface, can be used with the SPI watchdog timer (SPI WDT). This interface requires only one additional line, a chip select. Figure 1 shows a possible configuration. As illustrated this connection requires no additional components. Sample code, provided in a later section, is written to support the hardware shown in Figure 1.

Implementation

While the interface and code implementation is not complex, there are some areas where care must be taken to achieve functional code. A write enable command (WREN) must precede each write operation, including a write to the status register. The WREN command begins with the $\overline{CS}$ line going LOW and completed with the $\overline{CS}$ line returning HIGH. Once writes have been enabled, they are active only during a byte, page, or status register write. This means a WREN command must precede each write operation. The write enable bit is also reset automatically upon power-up.

It is possible to write a block of data in a single operation. However, each block is 32 bytes long and a block write cannot cross a block boundary. The block boundaries begin at addresses where bits A4 through A0 are “0”. As previously described, a WREN command is required before a new block can be written. This block write mechanism is implemented in the sample firmware code.

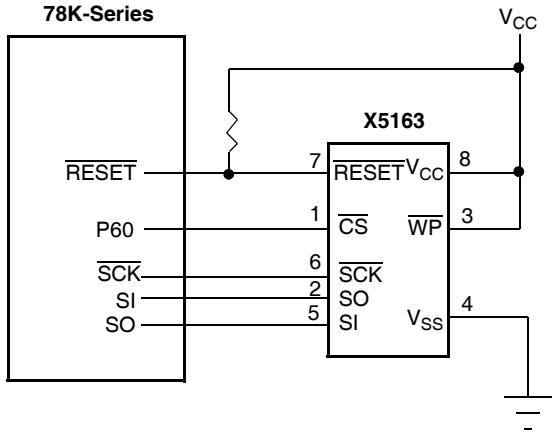

FIGURE 1. INTERFACING THE INTERSIL X5163 CPU SUPERVISOR TO THE 78K-SERIES µC USING THE SYNCHRONOUS SERIAL PORT

It is possible to write new values into the status register to change block protection and change the watchdog timer value. Since the status register is nonvolatile, a write to the register must follow the same restrictions as other nonvolatile writes. This means that a write to the status register will take a maximum of 10ms to complete, and cannot occur concurrently with data write operations.

When using the watchdog timer, a $\overline{RESET}$ signal is sent out after a selectable period of time. If the microcontroller does not respond in this amount of time, it will be reset. By toggling the $\overline{CS}$ line, the watchdog $\overline{RESET}$ can be held-off. The sample firmware code does not include this watchdog timer $\overline{RESET}$ hold-off operation.

The circuit of Figure 1 shows a pull-up resistor on the $\overline{RESET}$ line. This is required, since the SPI WDT has an open drain output. Typically, however, the microcontroller has a $\overline{RESET}$ circuit that allows a user initiated re-start. In this case, the resistor shown in the figure is not an additional component, but part of the reset mechanism.

Code Listings

The listing for the interface firmware is included on following pages. The code consists of a test program that moves a block of data from ROM to the EEPROM, then moves the block from EEPROM to the 78K2 internal RAM. The EEPROM-specific routines takes less than 170 bytes of code. These routines are:

Init_SIO—This routine will set up the synchronous serial port to communicate with the SPI WDT.

Put_Byte—This routine sends a data byte to the EEPROM using the internal hardware shifter of the microcontroller.

Write_Stat—This routine will write a value into the EEPROM status register.

Get_Byte—This routine gets a byte from the EEPROM using the internal hardware shifter of the microcontroller.

Wait_COM—After writing a byte to the microcontroller internal hardware shift register, this routine will wait for a byte transmit to complete.

Wait_EE—This routine will wait for a EEPROM write to complete.

E2_Command_Fix—This routine will send one of the various commands to the EEPROM.

Block_Read—This routine will read a block of data from the EEPROM and will save the block in RAM. The EEPROM source address pointer and the destination address pointer, along with the block size in bytes, are pre-specified.

Block_Write—This routine writes a block of data to the EEPROM. The data source address pointer and the EEPROM destination address pointer are pre-specified, as is the number of bytes in the block. This routine handles data blocks that do not begin on an EEPROM border and can handle blocks greater than 32 bytes.

Read_Stat—This routine will return the current value in the EEPROM Status register.

Conclusion

Few members of NEC's 78K-series microcontrollers come equipped with an on-board watchdog timer and only one family (the uPD7824x) has on-chip EEPROM. The introduction of the SPI WDT by Intersil removes these two limitations with a single 8-lead device. Since this combination requires no interface hardware and minimal code, it is the perfect combination for many industrial control applications.

Additional Intersil code can be found on the World Wide Web at http://www. intersil.com.

```
$       TITLE ('X25163Interface')
;       File Name: MPC
$       PC(213)
;_____________________________________
;
;       OPERATION:
;
;       This program will access the Intersil Serial EEPROM
;       with Serial Peripheral Interface (SPI) and watchdog timer
;
;       This routine is set up for an EEPROM that uses the
;       synchronous serial I/O port of the K-series devices.
;
;       This program is written for the DDB...
;_____________________________________
;       TESTPG:
;_____________________________________
;
;       Define Equates
;
WREN    equ     06H     ; Command: Write enable
WRDI    equ     04H     ; Command: Write Disable
RDSR    equ     05H     ; Command: Read Status Register
WRSR    equ     01H     ; Command: Write Status Register
READ    equ     03H     ; Command: Read EEPROM
WRITE   equ     02H     ; Command: Write EEPROM
;
WIP     equ     A.0     ; EEPROM Write in Progress
CE_     equ     P6.0    ; Chip enable line
Msg1    equ     5       ; Start address of EEPROM block
;
NUM_TRY equ     50      ; read WIP this long before giving up
;
;       Define Stack area
;
STKSEG  DSEG    AT  0FE00H
   DS      32
STACK:
;
;       Define Variables
;
VARIAB  DSEG    AT      0FE20H
BYTE_COUNT:     DS      1
MESSAGE:        DS      40      ; Where data is to be moved.
;                               ; Used for a test program..,
;
;       Vector Table
;
VRESET  CSEG    AT      9000H   ;
 BR      START           ;

CMAIN   CSEG    AT      9080H
;_____________________________________________
;       Main Routine
;_____________________________________________
;
;       Initialize System...
;
START:
   di                       ; Disable interrupts
   mov     MM, #00010111B   ; Ext ROM Fetch,no ext addr,1 wait
```

```
   mov     RFM, #00000000B    ; Disable refresh pulse out
   mov     PM6, #00000000B    ; Select Mem bank 0, P64-67=output
   movw    SP, #STACK         ; Set stack pointer
;
   call    !Init_SIO          ; Initialize Serial I/O port
;                                       Turn on Xmit & Recv
;
   mov     PM0, #0
   mov     P0, #0H
   set1    CE_                ; Disable the EEPROM
;_____________________________________________________________________
;
;       The following is a Test program that writes a block of
;       data into the EEPROM from ROM, then reads it back into
;       the 78K2 internal RAM area.
;_____________________________________________________________________
;
TEST:
   movw    HL, #MSG_ROM       ; Location of message in ROM
   movw    DE, #Msg1          ; Location of message 1 in EEPROM
   mov     BYTE_COUNT, #35    ; Write 35 bytes to EEPROM

   call    !Block_Write       ; Write the block

   movw    HL, #MESSAGE       ; Location of message in RAM
   movw    DE, #Msg1          ; Location of message 1 in EEPROM
   mov     BYTE_COUNT, #35    ; Read 35 bytes from EEPROM

   call    !Block_Read

   mov     X, #10H            ; Set WD Timer to 600 mSec
   call    !Write_Stat        ; Set WD Timer

   call    !Fini_SIO          ; Turn off Serial I/O port

LOOP:
   NOP
   BR      LOOP

MSG_ROM:
   DB      'This is a test of the Serial EEPROM'

;=====================================================================
;       Following are the various routines to complete the
;       above operation...
;
;       Init_SIO        Initialize the Serial I/O Port
;       Fini_SIO        Turn off the Serial I/O
;
;       Wait_COM        Wait for the communication to complete
;       Wait_EE         Wait for EEPROM Write to complete
;
;       Put_Byte        Sends one byte to the EEPROM
;       Get_Byte        Gets one Byte from the EEPROM
;
;       E2_Command_Fix  Sends one of 6 commands
;
;                       WREN (Write Enable)
;                       WRDI (Write Disable)
;                       RDSR (Read Status register)
;                       WRSR ( Write Status Register)
```

```
;                       READ (Read EEPROM)
;                       WRITE (Write EEPROM)
;
;       Read_Stat       Reads the EEPROM Status register
;       Write_Stat      Writes the EEPROM Status Register
;
;       Block_Read      Reads a block of data from the EEPROM
;       Block_Write     Writes a block of data to the EEPROM
;
;=====================================================================
;
;_____________________________________________________________________
;       Init_SIO
;
;       This routine will initialize the Serial I/O port for
;       Synchronous operation, using internal clocking at 750K bps.
;
;       Routines Called:        None
;       Input:                  None
;       Output:                 None
;       Registers used:         A
;_____________________________________________________________________
;
Init_SIO:

   or      MK0H, #10000000B ; Disable serial interrupt
   or      PMC3, #0CH       ; Use SO and SCK
   mov     CSIM, #2         ; Set Serial clock to fCLK/8
   set1    CTXE             ; Turn on transmit mode
   set1    CRXE             ; Turn on receive mode

End_Ser_Setup:

   clr1    WUP              ; Interrupt gen after each xfer
   clr1    CSIIF            ; Clear Sync Serial Intr Flag
   ret
;
;_____________________________________________________________________
;       Fini_SIO
;
;       This routine will initialize the Serial I/O port for
;       Synchronous operation, using internal clocking at 750K bps.
;
;       Routines Called:        None
;       Input:                  None
;       Output:                 None
;       Registers used:         A
;_____________________________________________________________________
;
Fini_SIO:

   clr1    CTXE            ; Turn off transmit mode
   clr1    CRXE            ; Turn off receive mode
   br      End_Ser_Setup
;_____________________________________________________________________
;       Wait_COM
;
;       This routine will wait for an interrupt to signal a xmit or
;       recv complete
;
;       Routines Called:        None
```

```
;       Input:                  None
;       Output:                 None
;       Registers used:         None
;_____________________________________________________________________
;
Wait_COM:

   btclr   CSIIF, $Return  ; Wait for completion of Xmit/Rcv
   br      Wait_COM

return:

 ret
;_____________________________________________________________________
;       Wait_EE
;
;       This routine will wait for the EEPROM write sequence to
;       complete.
;
;       Routines Called:        None
;       Input:                  None
;       Output:                 None
;       Registers used:         A, B
;_____________________________________________________________________
;
Wait_EE:

   mov     B, #NUM_TRY     ; Maximum number of samples

Wait_EE_LP:

   call    !Read_Stat      ; Read the Status Register

   bf      WIP, $Wait_done ; If Write complete, done...
   dbnz    B, $Wait_EE_LP  ; If not done, give it more time
                           ;       but not too much...
Wait_done:

 ret                    ; else, return
;_____________________________________________________________________
;       Put_Byte
;
;       This routine will move one byte of data from memory pointed
;       to by the HL register to the EEPROM.
;
;       Routines Called:        None
;       Input:                  HL = Address of data to send
;       Output:                 HL = Next address of data to send
;       Registers used:         A, HL, B
;_____________________________________________________________________
;
Put_Byte:

 mov     SIO, A          ; Put byte to serial port
 br      Wait_COM        ; Wait for byte to be sent
;_____________________________________________________________________
;       Get_Byte
;
;       This routine will move one byte of data from the EEPROM
;       to memory pointed to by the HL register.
;
```

```
;       Routines Called:        None
;       Input:                  None
;       Output:                 A = Returned byte
;       Registers used:         AX, B
;_____________________________________________________________________
;
Get_Byte:

   mov     SIO, #0          ; Send dummy byte to activate recv
   call    !Wait_COM        ; Wait for byte to be recv'd
   mov     A, SIO           ; Get byte
   ret
;_____________________________________________________________________
;       E2_Command_Fix
;
;       This routine will send a control signal to the EEPROM
;
;               06H     ; Command: Write enable
;               04H     ; Command: Write Disable
;               05H     ; Command: Read Status Register
;               01H     ; Command: Write Status Register
;               03H     ; Command: Read EEPROM
;               02H     ; Command: Write EEPROM
;
;       Routines Called:        None
;       Input:                  A = Command; DE = Address in EEPROM
;       Output:                 None
;       Registers used:         None
;_____________________________________________________________________
;

E2_Command_Fix:

   clr1    CE_              ; Enable EEPROM
   br      Put_byte         ; Write a Command to serial port

;_____________________________________________________________________
;       Read_Stat
;
;       This routine will read a value from the status register
;
;       Routines Called:        None
;       Input:                  None
;       Output:                 A = Status Reg value
;       Registers used:         A
;_____________________________________________________________________
;
Read_Stat:

   mov     A, #RDSR         ; Read Status Register
   call    !E2_Command_Fix

   call    !Get_byte        ;
   set1    CE_              ; Disable the chip

   ret
;_____________________________________________________________________
;       Write_Stat
;
;       This routine will write a value to the status register
;
```

```
;       Routines Called:        None
;       Input:                  X = Status register data
;       Output:                 None
;       Registers used:         A, X
;_____________________________________________________________________
;
Write_Stat:

   mov     A, #WREN         ; Prepare to enable writing
   call    !E2_Command_Fix  ; Send a WREN command
   set1    CE_              ; Disable EEPROM

   mov     A, #WRSR         ; Write Status Register
   call    !E2_Command_Fix

   mov     A, X             ; Write the status
   mov     P0, #1
   call    !Put_byte        ;
   set1    CE_              ; Disable the chip

   ret
;_____________________________________________________________________
;       Block_Read
;
;       This routine will read a block of data from the EEPROM
;
;       Routines Called:        Get_Data_Byte
;       Input:                  HL = Save address pointer,
;                               BYTE_COUNT = number of bytes
;       Output:                 HL = Address of next save location
;       Registers used:         A, HL, BYTE_COUNT
;_____________________________________________________________________
;
Block_Read:

   mov     A, #READ         ; Send reset command
   call    !E2_Command_Fix

   mov     A, D             ; Send upper address byte
   call    !Put_Byte        ; Send EEPROM Start Address
   mov     A, E             ; Send lower address byte
   call    !Put_Byte        ; Send EEPROM Start Address
                            ;   DE = EEPROM Address
Blk_Rd_Loop:

   call    !Get_Byte
   mov     [HL+], A
   dbnz    BYTE_COUNT, $Blk_Rd_Loop
   set1    CE_                      ; Disable EEPROM

   ret
;_____________________________________________________________________
;       Block_Write
;
;     This routine will write a block of data to the EEPROM
;     Since the EEPROM has a 32 byte page, a limit of 32 bytes of
;     data can be written before the issuing of a non-volatile write
;     cycle. Also, in order to avoid data wrapping on a page, care
;     must be taken when writing over page boundaries.
;
;       Routines Called: E2_Command_Fix, Put_Byte, Wait_EE
```

```
;       Input: DE = Internal Address of EEPROM
;                  (where data is to be written)
;                  HL = Address of data to be written
;                  BYTE_COUNT = Number of bytes to write
;       Output: None
;       Registers used: AX, DE, HL, BYTE_COUNT
;_____________________________________________________________________
;
Block_Write:

     mov    A, #WREN          ; Prepare to enable writing
     call   !E2_Command_Fix   ; Send a WREN command
     set1   CE_               ; Disable EEPROM

Write_OP:

     mov    A, #WRITE
     call   !E2_Command_Fix   ; Start writing
     mov    A, D              ; Send upper address byte
     call   !Put_Byte             ; Send EEPROM Start Address
     mov    A, E                  ; Send lower address byte
     call   !Put_Byte             ; Send EEPROM Start Address
                                  ; DE = EEPROM Address
Blk_Loop:

     mov    a, [HL+]              ; Get next byte
     call   !Put_Byte             ; Send it out
                                  ;  HL points to next byte
 dbnz BYTE_COUNT, $Next_bit       ; Count byte, if last one, go write
 br NV_Write                      ;   else check for 32 byte boundary

Next_bit:

   incw    DE             ; Increment EEPROM address pointer
   mov     A, E
   and     A, #31         ; Check for 32 byte block boundary
   cmp     A, #0          ; Is this a new block start?
   bne     $Blk_Loop      ; No, keep sending

NV_Write:

   set1    CE_               ; Disable EEPROM
   call    !Wait_EE          ; Wait for any writes to complete
   cmp     BYTE_COUNT, 0     ; If not all bytes are sent
   bne     Block_Write       ; keep on...
;
   mov     A, #WRDI
   call    !E2_Command_Fix   ; Disable writes
   set1    CE_               ; Disable EEPROM

 ret
 END
```

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.0