Application Note

Using Digital UART with the Z8F6423 MCU as Host

AN039601-1216

Abstract

This application note discusses the use of Zilog's single-chip Digital Universal Asynchronous Receiver/Transmitter (DUART) device as an additional serial communication device with the Z8F6423 microcontroller, a member of Zilog's Z8 Encore! XP F64xx Series of MCUs, as host.

Note: The source code file associated with this application note, AN0396-SC01, is available free for download from the Zilog website. This source code has been tested with ZDS II – Encore! version 5.2.2. Subsequent releases of ZDS II may require you to modify the code supplied with this application note.

Discussion

Zilog's Digital UART is suitable for use in applications where the host requires an additional asynchronous serial communication peripheral. This DUART chip can be included in a system and controlled via I^{2}C protocol (Two Wire Interface).

Communication

I^{2}C

DUART is an I^{2}C slave device using a 7-bit address and can support a maximum bus speed of 400 KHz. This device has up to eight possible addresses, allowing up to 8 devices on a single bus. The I^{2}C uses two bi-directional open-drain lines, pulled up to V_{DD} with resistors. All I^{2}C transactions must be separated by a wait period of at least four microseconds. Figure 1 shows the I^{2}C protocol.

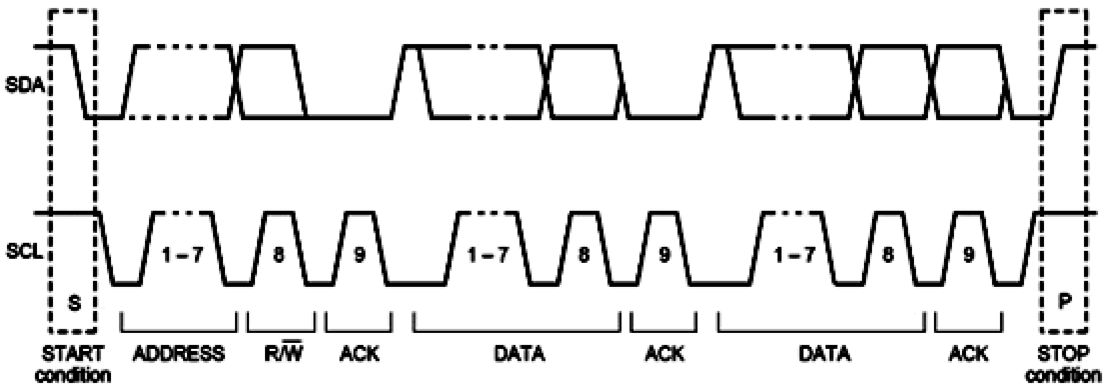


Figure 1. I^{2}C Protocol

I^{2}C Addressing

The DUART chip responds to the following addresses:

- `1010XXXb`, where `XXX` is the address configured using the I2CADDR pins. To be used to access the EEPROM through de facto standard interface
- `1011XXXb`, where `XXX` is the address configured using the I2CADDR pins. To be used to access the commands through standard I2C protocol

Commands

Commands are sent to communicate with the DUART. Table 1 list the commands and their description. The higher 3 bits of the command byte identify the peripheral. The EEPROM/GPIO peripheral uses `000b`, UART0 uses `001b`, UART1 uses `010b,` and SYSTEM uses `111b`. The command is the lower 5 bits of the command byte.

Table 1 lists the commands.

Table 1. Commands

Command Byte	Data Size (Bytes)	Direction	Peripheral	Description
0x00	1	Write	EEPROM	Write EEPROM
0x01	1	Read	EEPROM	Read EEPROM
0x02	2	Write	EEPROM	Write Current Location Register
0x03	2	Read	EEPROM	Read Current Location Register
0x04	1	Write	EEPROM	Erase Requested Page
0x06	2(4)[1]	Write	GPIO	Setting GPIO OUT Register
0x07	1(2)[1]	Read	GPIO	Reading GPIO IN Register
0x08	3(5)[2]	Write	GPIO	Write GPIO Configuration
0x09	3(5)[2]	Read	GPIO	Read GPIO Configuration
0x0F	1	Read	GPIO	Read GPIO Interrupt Status Register
0x21	1	Read	UART0	Read UART Status Register
0x22	1	Write	UART0	Enable Interrupts
0x23	1	Read	UART0	Interrupt Status Register
0x24	1	Write	UART0	Write Data to TX FIFO
0x25	1	Read	UART0	Read RX FIFO
0x26	2	Write	UART0	Write Baud Rate Register
0x27	2	Read	UART0	Read Actual Baud Rate Register
0x28	2	Write	UART0	Write Configuration
0x29	2	Read	UART0	Read Configuration
0x2A	1	Write	UART0	Write Transmit Watermark Register
0x2B	1	Read	UART0	Read Transmit Watermark Register
0x2C	1	Write	UART0	Write Receive Watermark Register
0x2D	1	Read	UART0	Read Receive Watermark Register
0x2E	1	Write	UART0	Enable UART
0x31	2	Read	UART0	Read Receive and Transmit FIFO Level Registers
0x41	1	Read	UART1	Read UART Status Register
0x42	1	Write	UART1	Enable Interrupts
0x43	1	Read	UART1	Interrupt Status Register
0x44	1	Write	UART1	Write Data to TX FIFO[3]
0x45	1	Read	UART1	Read RX FIFO Data
0x46	2	Write	UART1	Write Baud Rate Register
0x47	2	Read	UART1	Read Actual Baud Rate Register
0x48	2	Write	UART1	Write Configuration
0x49	2	Read	UART1	Read Configuration

Table 1. Commands (Continued)

Command Byte	Data Size (Bytes)	Direction	Peripheral	Description
0x4A	1	Write	UART1	Write Transmit Watermark Register
0x4B	1	Read	UART1	Read Transmit Watermark Register
0x4C	1	Write	UART1	Write Receive Watermark Register
0x4D	1	Read	UART1	Read Receive Watermark Register
0x4E	1	Write	UART1	Enable UART
0x51	2	Read	UART1	Read Receive and Transmit FIFO Level Register
0xE1	1	Read	SYSTEM	Read System Status Register
0xE3	1	Read	SYSTEM	Read Last Operation Result Register
0xE5	3	Read	SYSTEM	Read System Version
0xEF	1	Read	SYSTEM	Read Interrupt Source Register

Notes:

1. 1 ZDU0110QUX device uses 4 bytes; other devices use 2 bytes.
2. A command consists of a sub-command and data. The ZDU0110QUX device uses 5 bytes while other devices use 3 bytes. Sub-command `0x0A` uses only 1 byte.
3. Not allowed to be stacked.

DUART Write

Figure 2 shows how to communicate with the DUART to write on certain registers.

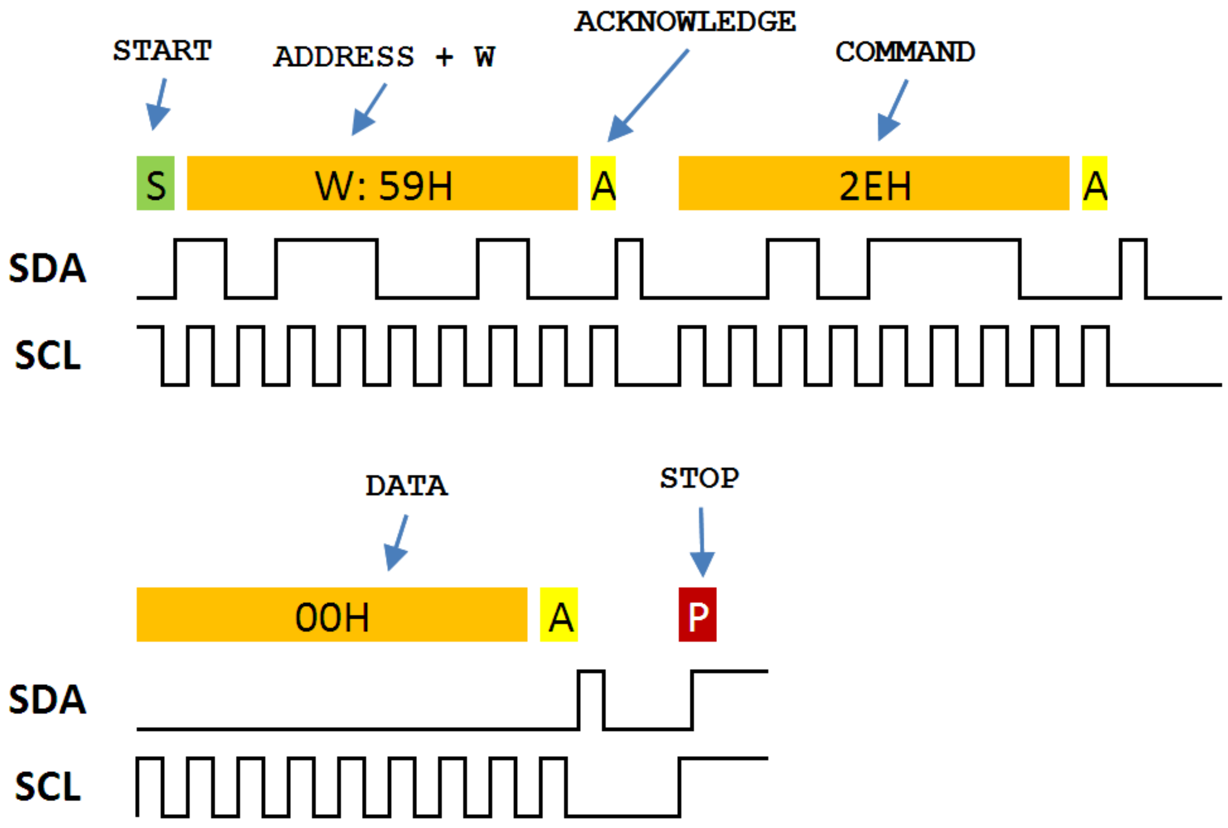


Figure 2. DUART Write

Communication starts with the host sending a Start condition, followed by the DUART address and write (0) bits. The host then sends the command byte and the required corresponding data. To ensure that the DUART is responding, an ACK bit should be read from the line after each byte is shifted. After sending all the information, the host issues a Stop condition.

DUART Read

Figure 3 displays the flow for reading data from the DUART.

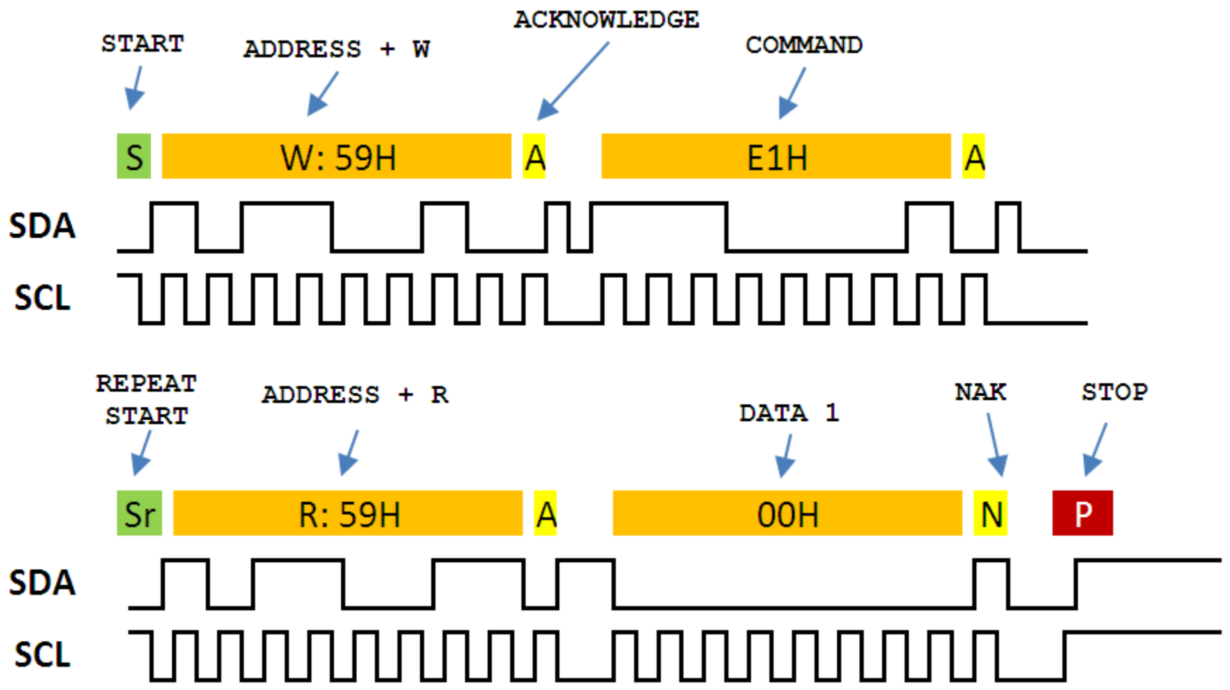


Figure 3. DUART Read

Communication begins with the host sending a Start condition, which is followed by the address and write bits. The DUART is then expected to reply with an ACK bit. It does so each time a byte is shifted to the line. Next, the host sends the command byte. Instead of issuing a Stop condition after the command byte, the host sends another Start condition (repeated start) followed by the DUART address and a read (1) bit. This time, the DUART sends the data for the host to receive. The host issues a NACK and the Stop condition after it receives the necessary data.

Stacked Write Commands

The DUART supports stacked write requests for multiple commands at the same time (up to a 64-byte packet). A stacked packet allows the Host to use one transaction to send multiple write commands, such as when configuring UART and/or GPIOs. If there is an error in the packet, processing is stopped and the error condition is logged in the System Status Register until the next request is processed.

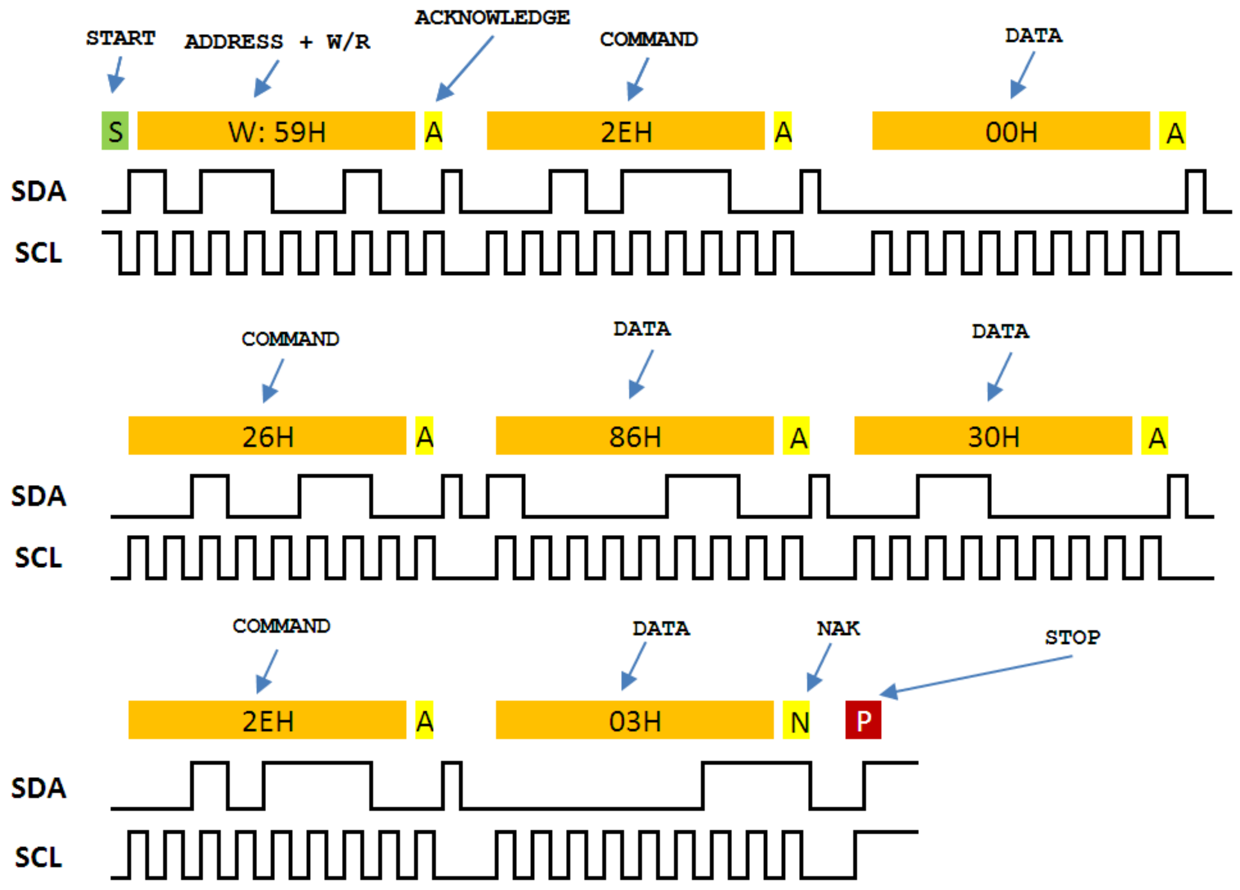


Figure 4. Stacked Write Commands

Software Details

The software for this application note is written in a modular way to allow users to easily copy and use the routines in their own application without modification. Table 2 lists the code files used in this application and a description of each file.

Table 2. Source Code Files

Source File	Description
main.c	Contains the main function of the software
i2c.c	Handles the routines regarding the I2C peripheral
eeprom.c	Handles all the routines that enable the host to access the EEPROM of the DUART
systems.c	Contains the routines that read the system status of the DUART
uart.c	Contains the routines regarding the UART functionality of the DUART
gpiox.c	Contains the routines regarding the GPIO functionality of the DUART
demo.c	This file contains the demo routines

Functions

Table 3 lists the functions and a brief description categorized by source code file.

Table 3. Source Code Files – Functions

Function Name	Source File	Description
main	main.c	Entry point of the program
delay	main.c	Simple for-loop routine for delay
i2c_write	i2c.c	Write to I2C Slave (DUART)
i2c_read	i2c.c	Read from I2C Slave (DUART)
i2C_readack	i2c.c	Function that waits for ACK from the slave
i2c_init	i2c.c	Initializes I2C Peripheral
eeprom_write	eeprom.c	Write to DUART EEPROM (de facto)
eeprom_read	eeprom.c	Read from DUART EEPROM (de facto)
eeprom_currentread	eeprom.c	Read EEPROM location pointer (de facto)
eeprom_write_I2C	eeprom.c	Write to EEPROM location pointer (I2C)
eeprom_set_loc_I2C	eeprom.c	Set EEPROM location pointer (I2C)
eeprom_read_I2C	eeprom.c	Read from EEPROM location pointer (I2C)
eeprom_read_loc_I2C	eeprom.c	Checks if the location pointer is in the right place
eeprom_erase	eeprom.c	Erase EEPROM (I2C)
edelay	eeprom.c	Simple for-loop routine for delay
system_readstat	systems.c	Read system status
system_readlastop	systems.c	Read last operation register
system_readsysver	systems.c	Read system version
system_IntSource	systems.c	Read interrupt source register
uart_readstatreg	uart.c	Read UART Status Register
uart_eninterrupts	uart.c	Enable interrupts for UART
uart_readintstat	uart.c	Read interrupt status
uart_writetxfifo	uart.c	Write data to TX FIFO
uart_readrxfifo	uart.c	Read data from RX FIFO
uart_writebrg	uart.c	Write Baud Rate Register
uart_readbrg	uart.c	Read Actual Baud Rate Register
uart_writeconfig	uart.c	Write UART configuration
uart_readconfig	uart.c	Read UART configuration
uart_writewtrmrk	uart.c	Write Watermark Register
uart_readwtrmrk	uart.c	Read Watermark Register
uart_enable	uart.c	Enable UART
uart_disable	uart.c	Disable UART
uart_readfifolvl	uart.c	Read RX and TX FIFO level

Table 3. Source Code Files – Functions (Continued)

Function Name	Source File	Description
uart_init	uart.c	Initialize UART
uart_printf	uart.c	DUART Data Out
uart_getstring	uart.c	Gets a definite number of characters and packs it as a string
udelay	uart.c	Simple for-loop routine for delay
gpio_setoutreg	gpiox.c	Request to set specific GPIO Out pins
gpio_readinreg	gpiox.c	Reads the current value on the GPIO pins for Input
gpio_writeconfig	gpiox.c	Sets GPIO configuration
gpio_readgpioconfig	gpiox.c	Read GPIO configuration
gpio_readgpioint	gpiox.c	Read GPIO interrupt
demo	demo.c	
demo_gpio	demo.c	GPIO demo routines
demo_eeprom	demo.c	EEPROM demo routines
demo_cnvrttoascii	demo.c	Converts a byte to ASCII
demo_cnvrfrmascii	demo.c	Converts an ASCII input to a byte

DUART Start-up Details

Upon power-up, the Digital UART device reads the I^{2}C addresses for the correct configuration and addressing. The system then asserts all interrupt pins, configures the I^{2}C host interfaces, configures all the peripherals to the default configurations, and then de-asserts all interrupts, notifying the host that the initialization is completed. Communication is not possible while the interface is being configured; however, after the host interface has been configured, the system will respond to a system status command while the rest of the system is being initialized.

Testing

This section discusses the procedure for testing the software and demonstrating this application.

Hardware Setup

Figure 5 shows the application hardware connections.

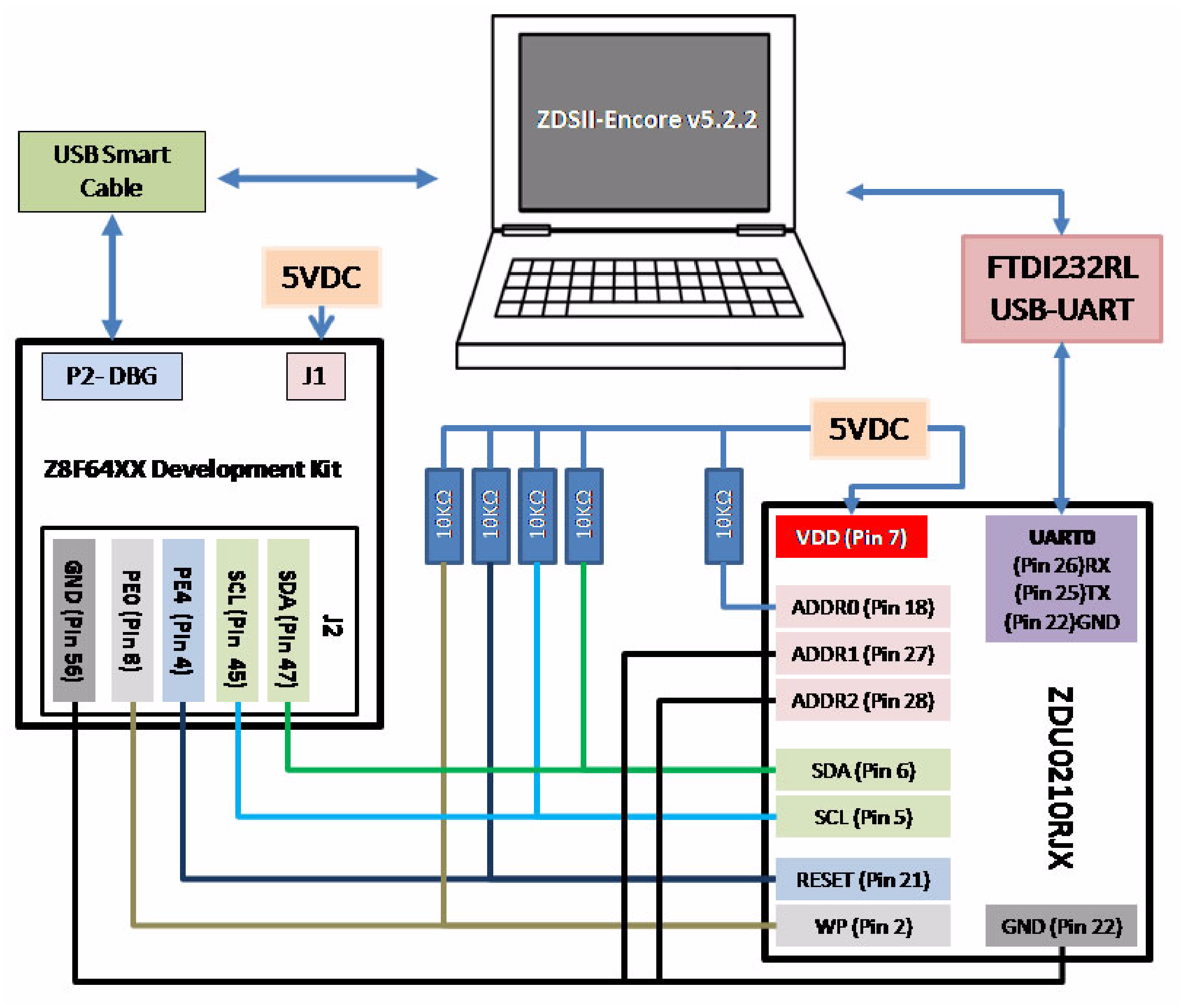


Figure 5. Hardware Setup

Software Setup

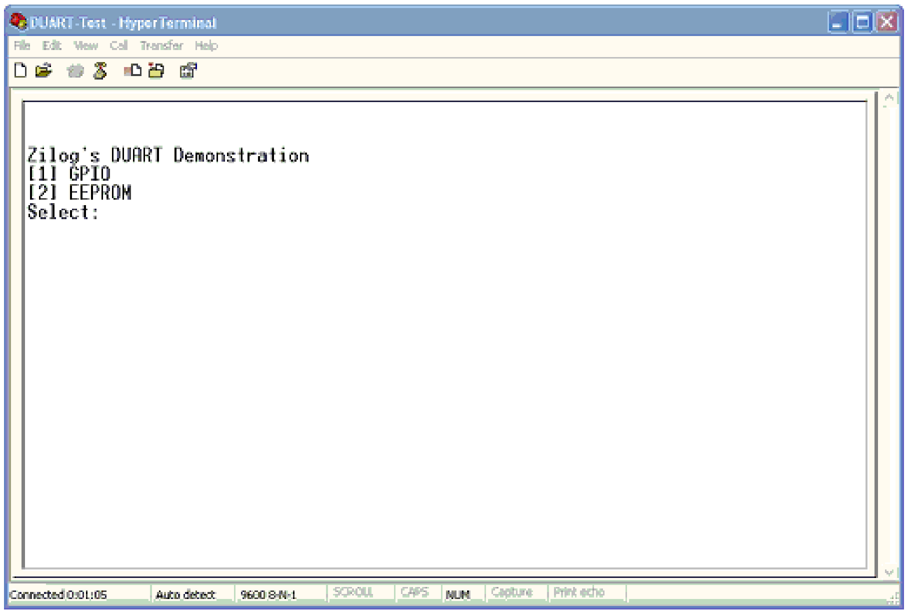
To install, configure, and test the software for this application, observe the following procedure:

1. Download ZDSII- Z8 Encore! Version 5.2.2 (or newer) from the Zilog Store and install it onto your PC.
2. Download the AN0396-SC01.zip source code file from the Zilog website and unzip it to an appropriate location on your PC.
3. Launch ZDSII-Z8 Encore! From the **File** menu, select **Open Project**.
4. Browse to the directory on your PC into which you downloaded the AN0396-SC01 source code. Locate the `AN0396_SC01.zdsproj` file and double-click to open.
5. Power up the MCU by supplying the 5 V DC power required by the Z8F6423 Development Kit.
6. Select **Rebuild All** from the **Build** menu to compile and flash the firmware to the Z8F6423 Development Kit.
7. Select **Debug** → **Download code** to flash the code to the MCU.
8. Wait for the code to be downloaded and then select **Debug** → **Stop Debugging**. At this point, the MCU is already loaded with the application firmware.
9. Power down the Z8F6423 Development Kit, and then disconnect the USB SmartCable.

Demonstration

Observe the following procedure for a demonstration of how this application works:

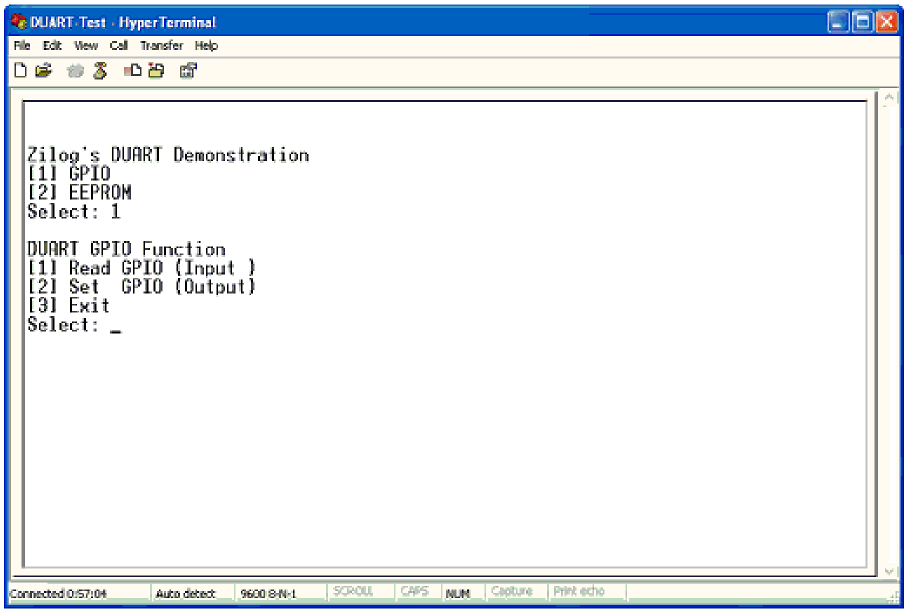
1. Power up the complete system.
2. Open HyperTerminal or any equivalent terminal emulation program. Configure it to 9600 baud, 8 bits data frame, no parity bits, and 1 stop bit.
3. In the HyperTerminal main menu, navigate to **File** → **Properties** → **Settings**. Click **ASCII Setup**, then select the **Echo typed characters locally** checkbox. Click **OK**.
4. Reset the MCU by pressing the reset switch on the Development Board.
5. After the reset, HyperTerminal displays the start-up menu, as shown in Figure 6, indicating that the MCU and the DUART are properly initialized.


DUART-Test - HyperTerminal
File Edit View Call Transfer Help
Connected 0:01:05 | Auto detect | 9600 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

```
Zilog's DUART Demonstration
[1] GPIO
[2] EEPROM
Select:
```

Figure 6. Terminal Display after Reset

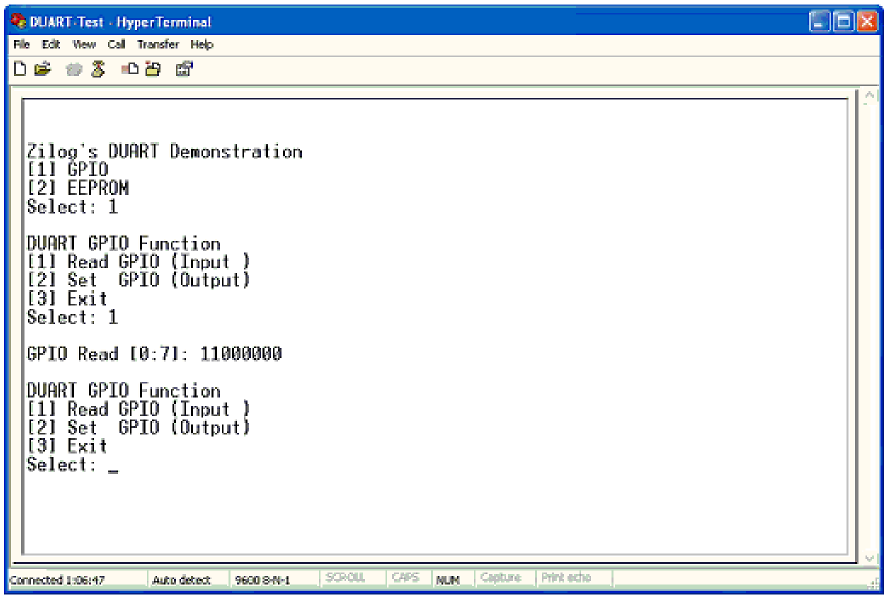
6. Enter the number corresponding to the functionality you want to use. Enter **1** for GPIO or **2** for EEPROM functionality.
7. Figure 7 shows the HyperTerminal display when option 1 (GPIO) is selected. It offers two choices – 1 to read GPIO input state (input) and 2 to set GPIO (output).


DUART-Test - HyperTerminal
File Edit View Call Transfer Help
Connected 0:57:04 | Auto detect | 9600 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

```
Zilog's DUART Demonstration
[1] GPIO
[2] EEPROM
Select: 1

DUART GPIO Function
[1] Read GPIO (Input )
[2] Set  GPIO (Output)
[3] Exit
Select: _
```

Figure 7. DUART GPIO Function

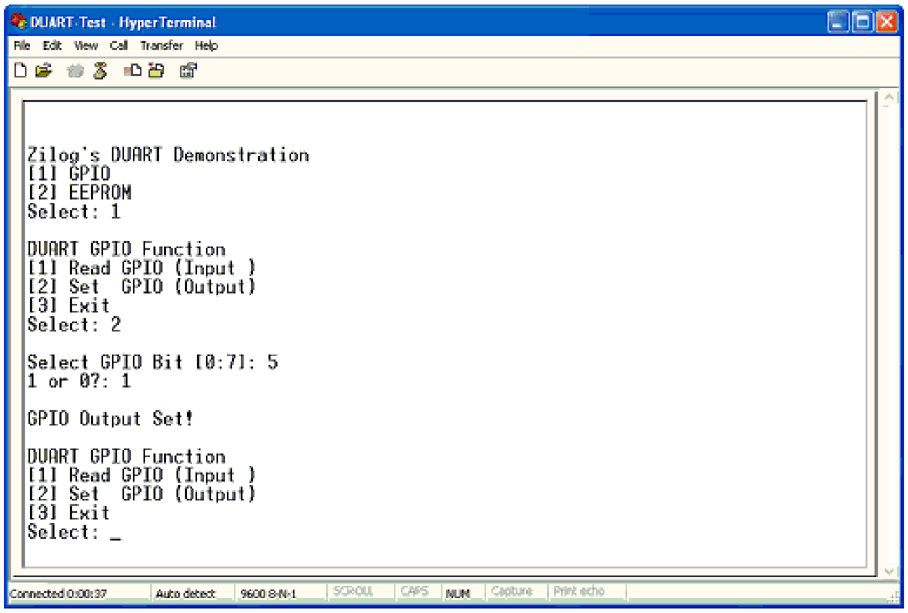
8. To read the GPIO state (input), enter **1**. Figure 8 shows the HyperTerminal display when this option is selected.


DUART-Test - HyperTerminal
File Edit View Call Transfer Help
Connected 1:06:47 Auto detect 9600 8-N-1 SCROLL CAPS NUM Capture Print echo

```
Zilog's DUART Demonstration
[1] GPIO
[2] EEPROM
Select: 1

DUART GPIO Function
[1] Read GPIO (Input )
[2] Set  GPIO (Output)
[3] Exit
Select: 1

GPIO Read [0:7]: 11000000

DUART GPIO Function
[1] Read GPIO (Input )
[2] Set  GPIO (Output)
[3] Exit
Select: _
```

Figure 8. DUART GPIO Read

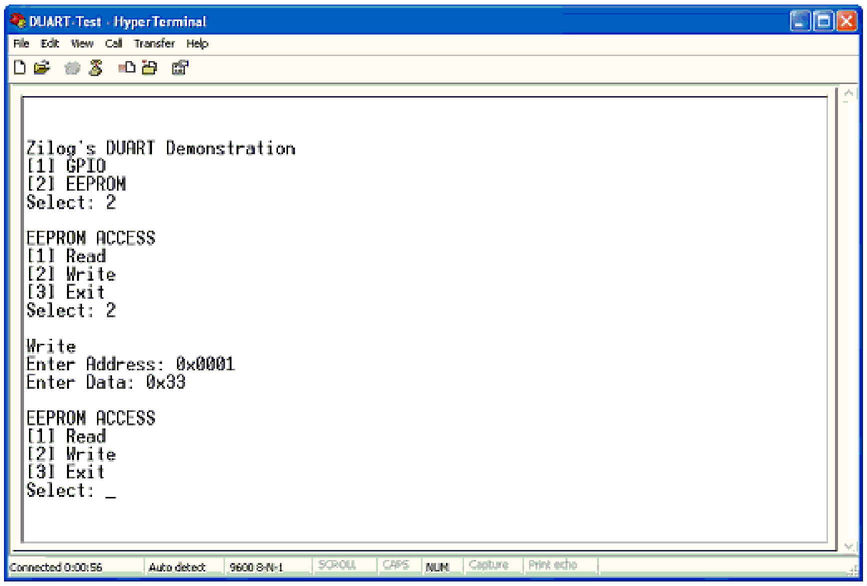
9. To set GPIO state (output), enter **2**. Then select the GPIO bit (0 to 7) and level (High (1) or Low (0)). Figure 9 shows the HyperTerminal display after making these selections.


DUART-Test - HyperTerminal
File Edit View Call Transfer Help
Connected 0:00:37 Auto detect 9600 8-N-1 SCROLL CAPS NUM Capture Print echo

```
Zilog's DUART Demonstration
[1] GPIO
[2] EEPROM
Select: 1

DUART GPIO Function
[1] Read GPIO (Input )
[2] Set  GPIO (Output)
[3] Exit
Select: 2

Select GPIO Bit [0:7]: 5
1 or 0?: 1

GPIO Output Set!

DUART GPIO Function
[1] Read GPIO (Input )
[2] Set  GPIO (Output)
[3] Exit
Select: _
```

Figure 9. DUART GPIO Write

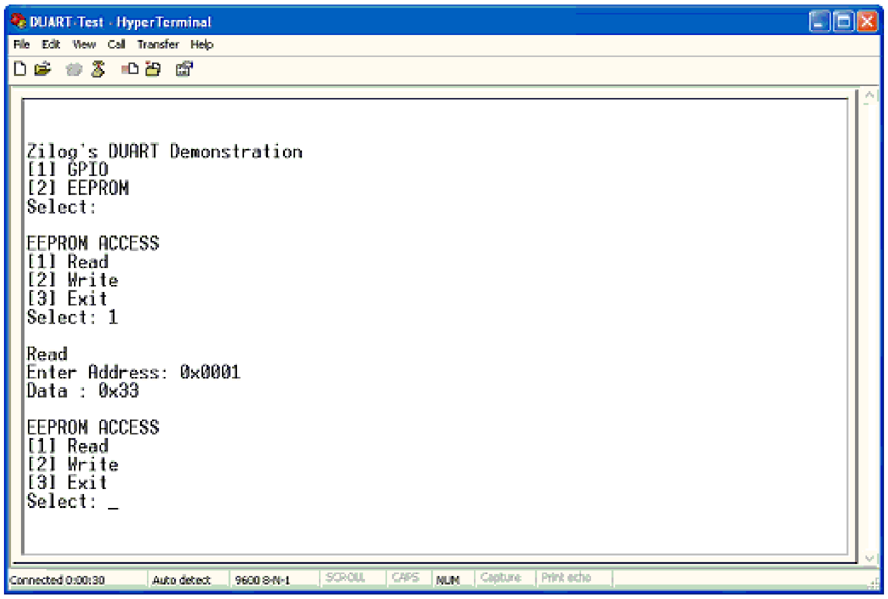
10. Enter **3** to exit the DUART GPIO function.
11. At the start-up menu, enter **2** to use the EEPROM functionality of the DUART. To write data in the EEPROM, enter **2**. Figure 10 shows the HyperTerminal display.



```
Zilog's DUART Demonstration
[1] GPIO
[2] EEPROM
Select: 2

EEPROM ACCESS
[1] Read
[2] Write
[3] Exit
Select: 2

Write
Enter Address: 0x0001
Enter Data: 0x33

EEPROM ACCESS
[1] Read
[2] Write
[3] Exit
Select: _
```

Figure 10. DUART EEPROM Write

12. Enter **1** to read the EEPROM. Enter the address (for example, `0x0001`) to display the 8-bit data stored in that location. Figure 11 shows the HyperTerminal display.



```
Zilog's DUART Demonstration
[1] GPIO
[2] EEPROM
Select:

EEPROM ACCESS
[1] Read
[2] Write
[3] Exit
Select: 1

Read
Enter Address: 0x0001
Data : 0x33

EEPROM ACCESS
[1] Read
[2] Write
[3] Exit
Select: _
```

Figure 11. DUART EEPROM Read

Equipment Used

This section provides a complete list of the hardware and software requirements for this application.

Hardware

Table 4 lists the hardware tools used to develop this application.

Table 4. Application Hardware

Description	Quantity
Z8F64200100KITG	1
ZDU0210RJX DUART	1
Zilog USB SmartCable	1
RS-232 Cable	1
UART to USB Converter FTDI232RL	1
5 VDC Adapter	1

Software

The software tools used to develop this application are:

- ZDSII – Encore 5.2.2
- AN0396-SC01.zip, containing the project file and source code files
- HyperTerminal or any equivalent communication and terminal emulation program

Summary

This application note discusses a methodology to interface a host MCU with Zilog's Digital UART chip through I^{2}C. This document also describes the use of Zilog's DUART as a serial peripheral extender to a host without sufficient GPIO pins to offer UART functionality. Additionally, Zilog's DUART provides extra GPIO pins and memory (EEPROM), which users can utilize to add features to their applications.

References

Documents associated with this application note are listed below. Each of these documents can be obtained from the Zilog website by clicking the link associated with its document number.

- Z8F6423 Development Kit User Manual (UM0151)
- Z8F64XX Product Specification (PS0199)
- Zilog DUART Product Specification (PS0389)

Appendix A. Schematic Diagram

Figure 12 shows a schematic diagram of the 28-pin DUART device.

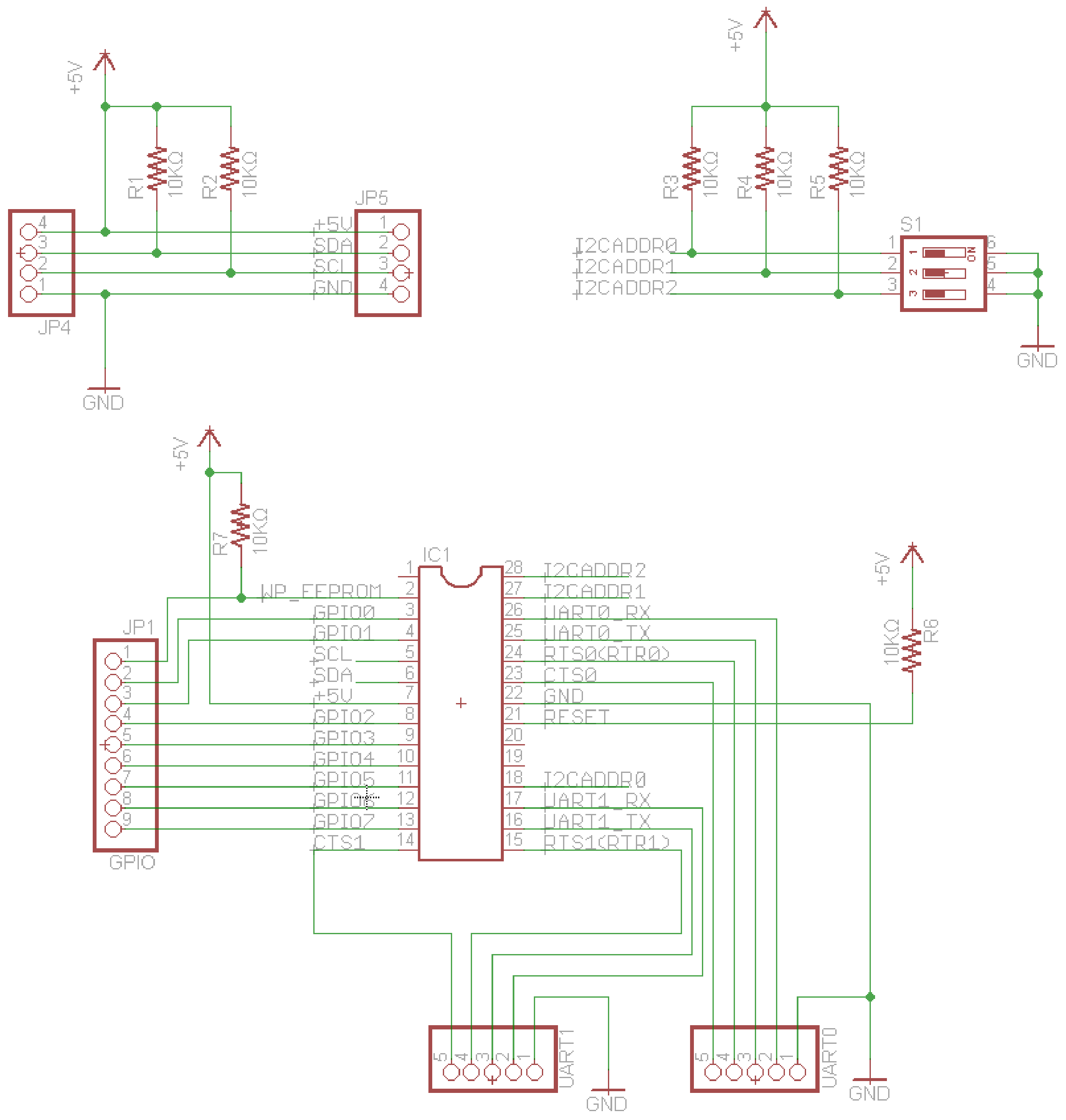


Figure 12. 28-Pin DUART Schematic Diagram

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.