User's Manual for
I2C/UART
Evaluation Board

September 27, 1999

1.0 Introduction

The Scenix I^{2}C/UART demo board provides an ideal platform for quick development and testing of I^{2}C and UART serial interface implementations. This platform allows the user to program two SX28AC microcontrollers, which are interconnected through an I^{2}C bus. The I^{2}C/UART demonstration software exercises many of the features of the demo board, whilst providing a good foundation for additional design expansion. The software utilizes the I^{2}C and UART Virtual Peripheral™ modules, developed at Scenix to demonstrate the flexibility of the SX and easy integration into various applications.

The documentation software assumes the user is familiar with the I^{2}C concept. For more details on I^{2}C interface, see the Philips' I^{2}C bus specification.

2.0 Features

- Two in circuit programmable 50MIPS SX28AC microcontrollers
- Dual Multi-Master I2C and Dual UART implemented on a single SX28AC
- User selectable clock sources for 50MHz or 8MHz operation
- Two on board I2C busses
- Three serial ports for easy debugging
- Jumper pins for selection of demo modes
- On board I2C based EEPROM
- I^{2}C clock/calendar interface
- Prototype area (1.0” x 4.5”)

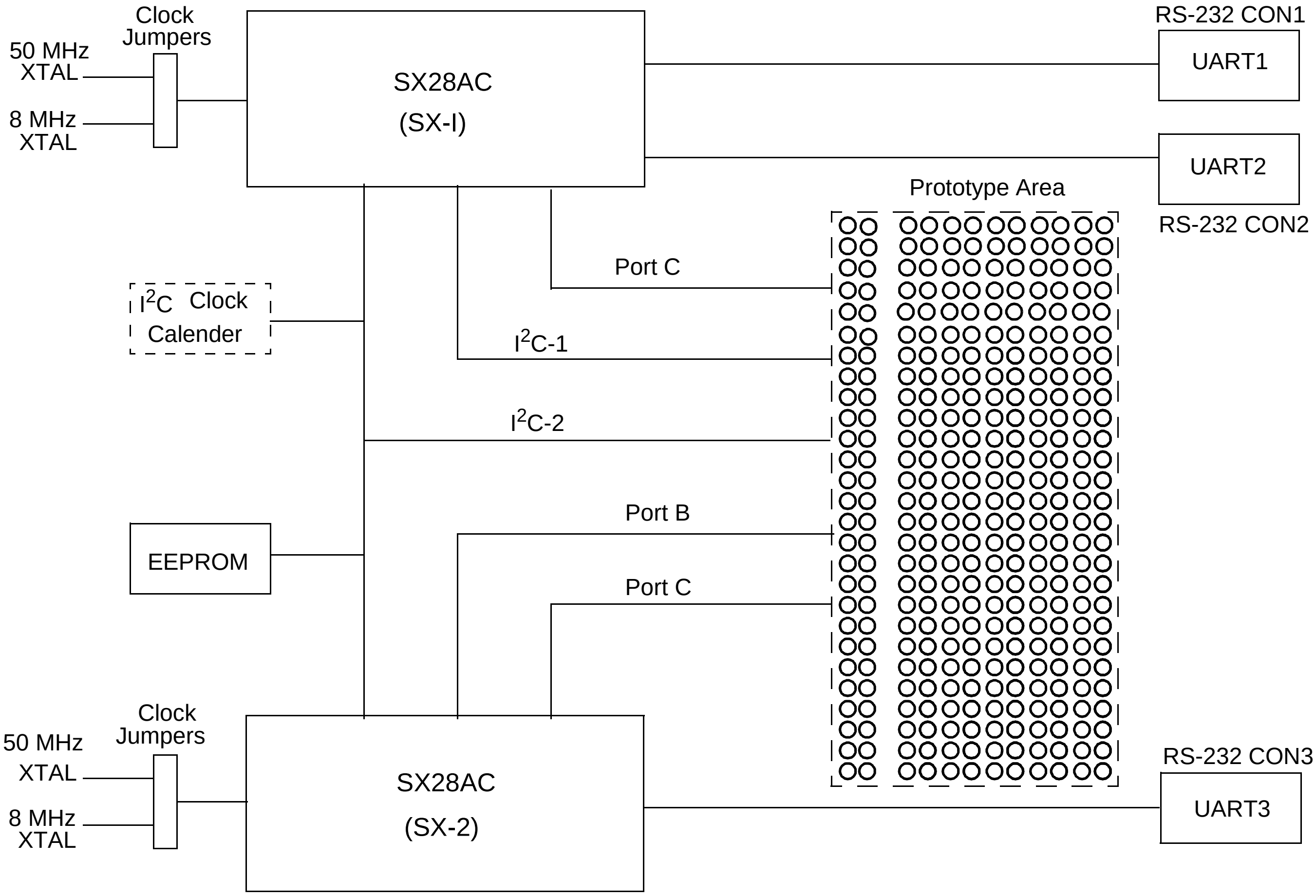


Figure 2-1. I2C/UART Demo Board Block Diagram

3.0 Description of Building Blocks

3.1 SX-2 (SX28AC-U2)

SX-2 is connected to a single I^{2}C bus and a single UART. By using jumper settings, the functionality of the SX can be changed. The SX-2 code can run either as a single I^{2}C master, single I^{2}C slave or as an I^{2}C multi-master and slave simultaneously. For more details about the software see Sec*tion 5.0.*

3.2 SX-1 (SX28AC-U1)

SX-1 is connected to both I^{2}C busses and two of the UARTs connectors. For SX-1, the Virtual Peripheral modules were integrated to run dual UARTs, dual I^{2}C masters and an I^{2}C slave simultaneously.

3.3 Clocking Options

Two clock sources are available for each SX28AC. Either 8 MHz or 50MHz operation are selectable by using the appropriate jumper settings. See S*ection 4.0* for description of jumper settings.

3.4 UART (Universal Asynchronous Receiver Transmitter)

SX-1 supports two independent UART interfaces while SX2 supports the third UART. The UARTs are useful to assist in debugging and as a user interface to the SX. When running the demo program, each UART is assigned an SX I^{2}C master. This allows the user to control each SX master directly.

3.5 Power Supply

The power supply circuitry utilizes the LM317 adjustable voltage regulator to a supply voltage of 2.5 to 5.5V. The supply voltage can be adjusted in this range by the potentiometer (R15). The default voltage setting is 5V.

3.6 EEPROM (Electrically Erasable Programmable Read Only Memory)

The I^{2}C/UART demo board contains the Microchip 24LC16B I^{2}C EEPROM. This allows direct access to 256x8 bit memory with the 2-wire I^{2}C interface. For more details, see the 24LC16B datasheet. The demo software written will execute the simple read and write routines to access the EEPROM. The prototype area can be used to easily add another EEPROM to the other I^{2}C bus.

3.7 Clock/Calendar

A Philips PCF8573 Clock/calendar interface and footprint is provided on the I^{2}C/UART demo board. This gives the user the ability to use the on-board PCF8573 device and test a multi-master system. The PCF8573 is not placed on the board.

3.8 I^{2}C Bus

The I^{2}C/UART demo board has two I^{2}C bus interfaces. I^{2}C-1 bus simply connects the SX-1 device to a pin header located adjacent to the prototype area where I^{2}C devices can be easily added and tested. I^{2}C- 2 bus connects SX-1 to SX-2 as well as connecting the EEPROM device and the clock/calendar interface. The I^{2}C-2 bus is also connected a pin header for easy addition of I^{2}C devices within the prototype area. As required by the I^{2}C specification, each bus contains the SDA (Serial Data Line), the SCL (Serial Clock Line), and pull-up resistors on each line. Configuring the appropriate SX I/O pin as an output or high impedance input changes the logic level on each line since the port acts as an open collector.

3.9 Prototype Area

The prototype area provides the user design expansion to implement additional circuitry. Headers adjacent to the prototype area provide easy access to all I/O SX-1/SX-2 I/O ports, I^{2}C busses, and clock/calendar interface.

4.0 Jumper Settings

4.1 Power Enable Jumpers

JP1 & JP2 Power and ground for SX-1, respectively. Connect both of these jumpers if SX-1 is required to be active.

JP4 & JP3 Power and ground for SX-2, respectively. Connect both of these jumpers if SX-2 is required to be active.

4.2 SX-2 Options (SX28AC-U2)

For all of the jumper settings listed below, a jumper placed across the pins will ground the respective I/O pin of the SX.

MSTR Selects SX-2 to run as a single master

SL Selects SX-2 to run as a single slave

MM Selects SX-2 to run the master in multi-master mode. This means that the arbitration algorithm is activated. Ensure that the MSTR jumper is also connected when running in multi-master mode.

8/50 When running the SX at 50Mhz this jumper must be set so that the correct reload values for ISR times and baud rates are used.

4.3 SX-1 Options (SX28AC-U1)

MODE Not used.

8/50 When running the SX at 50Mhz this jumper must be set so that the correct reload appropriate values for Interrupt Service Routine (ISR) timing and baud rates.

4.4 Clock Options

Jumpers 13 to 16 are used to select the operating frequency of each SX device.

SX 1 Clock Selction

	50Mhz	8Mhz
JP13	1-2	2-3
JP14	1-2	2-3

SX 2 Clock Selection

	50Mhz	8Mhz
JP15	1-2	2-3
JP16	1-2	2-3

5.0 Demo Software

5.1 Introduction

There are two main demo programs for the SX I^{2}C /UART board, one for each SX. For SX-1, the source code is `I2C_uart demo1.src` and for SX-2 the source is `I2C_uart demo2.src`. Both of these programs have been derived the Virtual Peripheral™ modules. The Virtual Peripheral modules which have been used are I^{2}C Slave, I^{2}C master and UART. For a more in depth description of the individual modules see the appropriate application notes available at www.scenix.com.

The Virtual Peripheral modules are implemented in such a manner to appear as the user is using an hardware peripheral block. This means that the I^{2}C and UART operations are hidden from the main program logic, and all the user needs to do is call an access subroutine, and it can proceed to perform other tasks while the I^{2}C masters and slaves do their work. When finished, the I^{2}C devices will set flags to indicate whether or not the communications were successful.

5.2 SX Demo Software Description

The SX-2 demo software has been written to demonstrate the SX running as an I^{2}C master or slave or both simultaneously. The SX-1 software however is a bit more advanced as it runs dual I^{2}C masters and dual UARTs. Both programs have been written from the same basic code and therefore they look very similar. In order to run dual master and dual UARTs the SX-code uses two extra register banks to store all the required variables. This means that the only difference between the SX-1 and SX-2 codes is that there are more bank switches required to change between the active master and UART. To test the software, follow see quick start guide.

The software is written in three main sections, the interrupt service routine (ISR), mainline code and mainline subroutines.

5.3 The Interrupt Service Routine (ISR)

The ISR handles all the timing for the UART and I^{2}C operations. The ISR is responsible for maintaining the current state of each peripheral and running the state machines. The ISR will set flags when data has been sent or received and update the I/O pins. Most of the action within the ISR is transparent to the mainline code and little or no modification is required when changing the operation of the program. Possible modifications include changing the interrupt frequency and removing or adding more software peripherals. The ISR executes four main blocks of code in the order shown below:

- I^{2}C master state machine is executed
- I^{2}C slave state machine
- UART is updated, bit sent or received
- I^{2}C master arbitration is done if multi-master is selected

The arbitration is done after the UART code to allow time for the I^{2}C I/O pins to settle.

Each one of the above routines are described in detail in their respective application note available at www.scenix.com
A flow chart of the SX-2 ISR is shown in Figure 5.1.

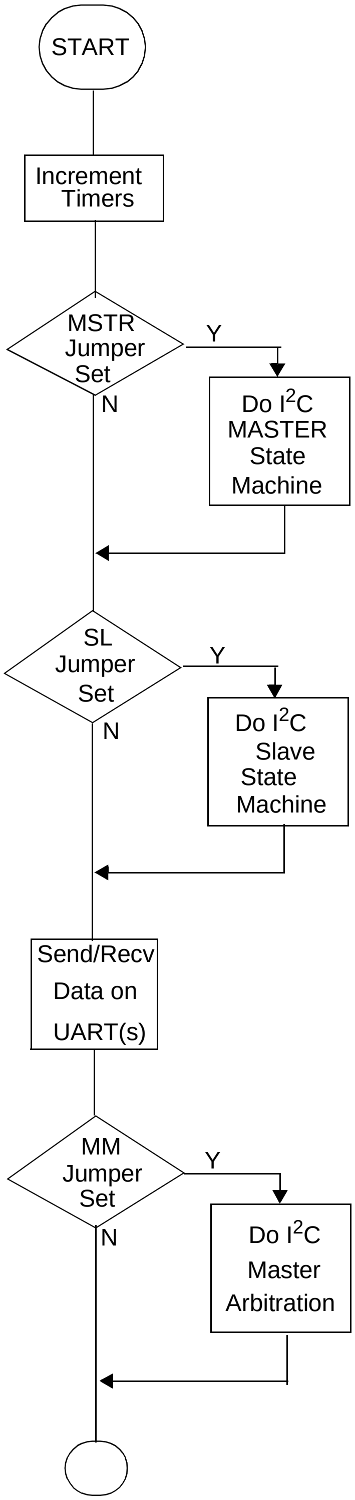

START
Increment Timers
MSTR Jumper Set
Y
N
Do I2C MASTER State Machine
SL Jumper Set
Y
N
Do I2C Slave State Machine
Send/Recv Data on UART(s)
MM Jumper Set
Y
N
Do I2C Master Arbitration

Figure 5-1. SX 2 Interrupt Service Routine Flow Diagram

5.4 Mainline Code

The mainline code controls the flow and functionality of the program. For this demonstration software, there are six main blocks associated with the mainline code:

- Initialize I/O and memory
- Detect and display all slaves on the I^{2}C bus
- Display instructions and get a command from the user
- Save a string to the EEPROM
- Read a string from the EEPROM
- Respond to slave being read

5.5 Mainline Subroutines

The mainline subroutines have been written to perform I^{2}C and UART operations by simply calling a subroutine. For example, to send a string to a UART, simply change to the UART bank you wish to send the string, load W (working register) with the address of the string and call ‘send_string’. The following routines have been written to simplify UART and I^{2}C operations:

UART:

• Get_byte	Reads a byte from the active UART
• Send_byte	Sends a byte on the active UART
• Send_string	Sends a string to the active UART
• Uppercase	Converts a character to uppercase
• Send_hex	Sends a number in hex to the active UART
• Get_hex	Gets a number in hex from the active UART

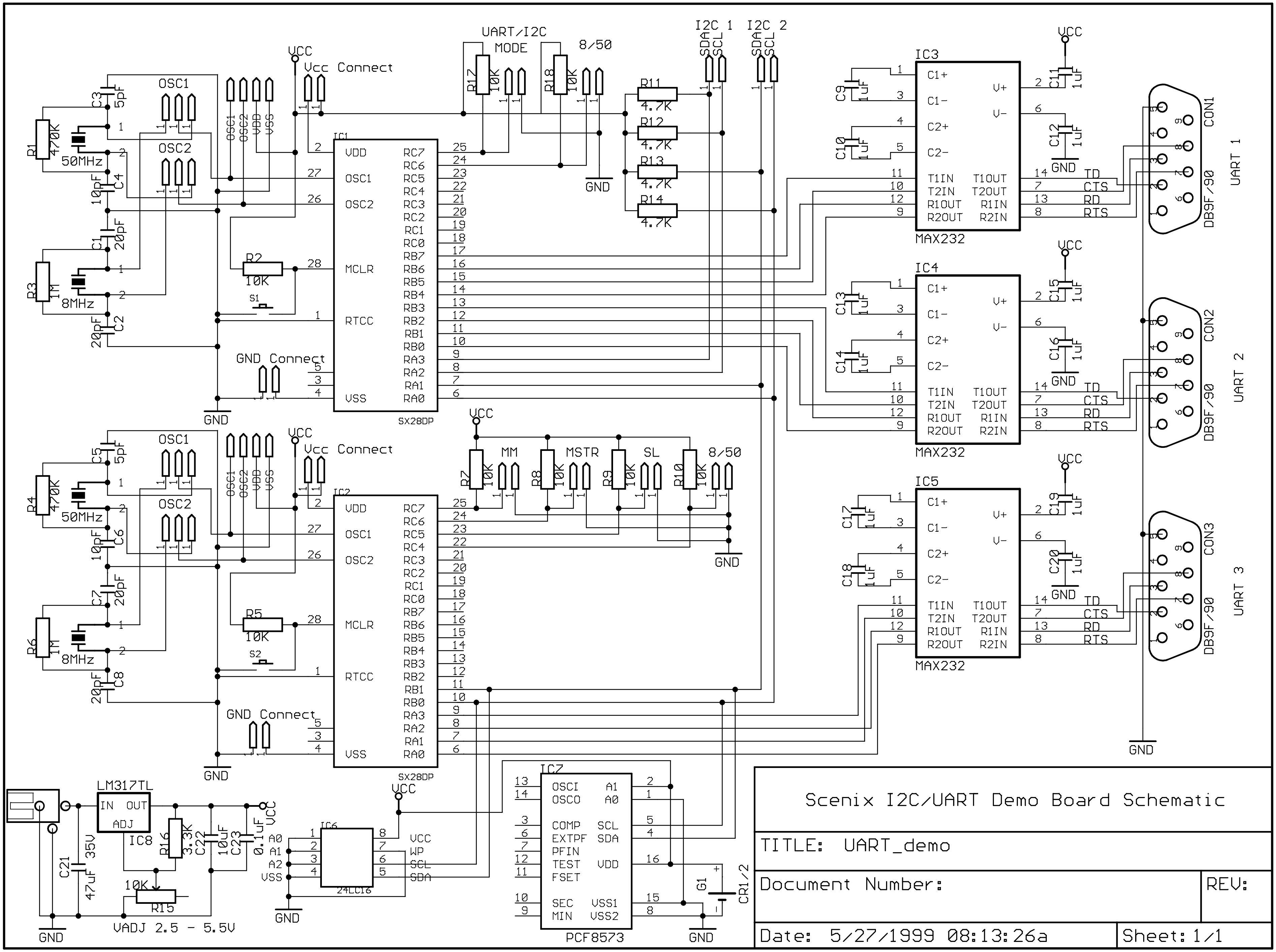
I^{2}C Master:

• I2CM_send_byte	Sends a byte from the active I^{2}C master
• I2CM_send_bytes	Sends multiple bytes from the active I^{2}C master
• I2CM_wait_not_busy	Waits for the I^{2}C master state machine to return to idle
• I2CM_get_byte	Gets a byte from an I^{2}C slave device
• I2CM_get_bytes	Receives multiple bytes from an I^{2}C slave device

EEPROM:

• EE_read_char	Reads a character from an EEPROM slave using active I^{2}C master
• EE_write_char	Writes a character to an EERPOM slave using active I^{2}C master

See the documentation within the source code for an explanation of each individual function.



Scenix I2C/UART Demo Board Schematic

TITLE: UART_demo

Document Number:

REV:

Date: 5/27/1999 08:13:26a

Sheet: 1/1