



PICkit[™] Serial Analyzer USER'S GUIDE

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, KEELOQ logo, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rfPIC, and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, PS logo, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2007, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. The Company's quality system processes and procedures are for its PIC[®] MCUs and dsPIC[®] DSCs, KEELOQ[®] code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

**QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949:2002 ==**

Table of Contents

Preface	1
Chapter 1. PICKit™ Serial Analyzer Overview	
1.1 Introduction	5
1.2 Highlights	5
1.3 PICKit™ Serial Analyzer Contents	5
1.4 PICKit™ Serial Analyzer Development System	5
1.5 PICKit™ Serial Analyzer Hardware	6
1.6 PICKit™ Serial Analyzer Software	8
Chapter 2. Getting Started	
2.1 Introduction	9
2.2 Highlights	9
2.3 Installing the PICKit™ Serial Analyzer Software	9
2.4 Connecting the PICKit™ Serial Analyzer to the PC	9
2.5 Connecting the PICKit™ Serial Analyzer to the 28-Pin Demo Board 10	
2.6 Starting the PICKit™ Serial Analyzer Program	10
2.7 Running the 28-Pin Demo I ² C™ Demonstration Program	11
2.8 I ² C™ Communications – Basic Operations	14
2.9 28-Pin Demo I ² C™ Source Code and Firmware	15
Chapter 3. PICKit™ Serial Analyzer PC Program	
3.1 Introduction	17
3.2 Highlights	17
3.3 Installing the PICKit™ Serial Analyzer Software	17
3.4 Starting the Program	17
3.5 Configuration Wizard	18
3.6 Main Window	21
3.7 Serial Communications Modes	25
Chapter 4. I²C™ Master Communications	

PICkit™ Serial Analyzer User's Guide

4.1	Introduction	27
4.2	Highlights	27
4.3	PICkit Serial Pin Assignments	27
4.4	Selecting Communications Mode	27
4.5	Configuring I ² C Communications Mode	28
4.6	Communications: Basic Operations	30
4.7	Script Builder	31
4.8	Script Execute	35

Chapter 5. I²C™ Slave Communications

5.1	Introduction	37
5.2	Highlights	37
5.3	PICkit Serial Pin Assignments	37
5.4	Selecting Communications Mode	38
5.5	Configuring I2C Slave Communications Mode	38
5.6	Communications: Basic Operations	41
5.7	Communications: Profile Generator	42

Chapter 6. Lin Communications

6.1	Introduction	44
6.2	Highlights	44
6.3	PICkit Serial Pin Assignments	44
6.4	Selecting Communications Mode	44
6.5	Configuring Lin Slave Communications Mode	45
6.6	Communications: Basic Operations	47

Chapter 7. SPI and Microwire Master Communications

7.1	Introduction	49
7.2	Highlights	49
7.3	PICkit™ Serial Analyzer Pin Assignments	49
7.4	Selecting Communications Mode	49
7.5	Configuring SPI Communications Mode	50
7.6	Communications: Basic Operations	52

Table of Contents

7.7	Script Builder	54
7.8	Script Execute	59
Chapter 8. USART Asynchronous Communications		
8.1	Introduction	63
8.2	Highlights	63
8.3	PICKit Serial Pin Assignments	63
8.4	Selecting Communications Mode	64
8.5	Configuring USART Asynchronous Communications Mode	64
8.6	Communications: Basic Operations	66
8.7	Script Builder	66
8.8	Script Execute	71
Chapter 9. USART Master Synchronous Communications		
9.1	Introduction	73
9.2	Highlights	73
9.3	PICKit Serial Pin Assignments	73
9.4	Selecting Communications Mode	73
9.5	Configuring USART Synchronous Master Communications Mode .. 74	
9.6	Communications: Basic Operations	76
9.7	Script Builder	76
9.8	Script Execute	81
Chapter 10. User Defined Templates		
10.1	Introduction	83
10.2	Highlights	83
10.3	Creating and Using Templates	84
10.4	My Templates	85
10.5	GRAPHING DATA	86
Chapter 11. PICKit™ Serial Analyzer Firmware		

PICkit™ Serial Analyzer User's Guide

11.1	Introduction	87
11.2	Highlights	87
11.3	Overview	87
11.4	EXEC	89
11.5	COMM	92
11.6	I ² CM Communications	96
11.7	I ² CS Communications	102
11.8	SPI Communications	112
11.9	USART Communications	115
11.10	LIN Communications	119
Chapter 12. PICkit™ Serial Analyzer DLL		
12.1	Introduction	127
12.2	Overview	127
Chapter 13. Troubleshooting		
13.1	Introduction	129
13.2	Frequently Asked Questions	129
Appendix A. PICkit Serial Analyzer Schematics		131
A.1	Introduction	131
Appendix B. 28-Pin Demo Board I²C™ Demonstration Firmware ...		135
B.1	Introduction	135
B.2	Highlights	135
B.3	Hardware	135
B.4	Firmware	135
B.5	I ² C Communications	136
B.6	Slave Devices	137
B.7	Functions	140
Worldwide Sales and Service		143

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a “DS” number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is “DSXXXXA”, where “XXXX” is the document number and “A” is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE on-line help. Select the Help menu, and then Topics to open a list of available on-line help files.

INTRODUCTION

This chapter contains general information that will be useful to know before using the PICKit™ Serial Analyzer User's Guide. Items discussed in this chapter include:

- Document Layout
- Conventions Used in this Guide
- Warranty Registration
- Recommended Reading
- The Microchip Web Site
- Development Systems Customer Change Notification Service
- Customer Support
- Document Revision History

DOCUMENT LAYOUT

This document describes how to use the PICKit™ Serial Analyzer as a development tool to communicate with embedded development systems via serial protocols. The manual layout is as follows:

- **Chapter 1: PICKit™ Serial Analyzer Overview**
- **Chapter 2: Getting Started**
- **Chapter 3: PICKit™ Serial Analyzer PC Program**
- **Chapter 4: I²C™ Master Communications**
- **Chapter 5: SPI Master Communications**
- **Chapter 6: USART Asynchronous Communications**
- **Chapter 7: USART Master Synchronous Communications**
- **Chapter 8: User Defined Templates**
- **Chapter 9: PICKit™ Serial Analyzer Firmware**
- **Chapter 10: PICKit™ Serial Analyzer DLL**

PICKit™ Serial Analyzer User's Guide

- Chapter 11: Troubleshooting
- Appendix A: Hardware Schematics
- Appendix B: 28-Pin Demo Board I²C™ Demo Firmware

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	<i>MPLAB® IDE User's Guide</i>
	Emphasized text	...is the <i>only</i> compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	<u><i>File>Save</i></u>
Bold characters	A dialog button	Click OK
	A tab	Click the Power tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	#define START
	Filenames	autoexec.bat
	File paths	c:\mcc18\h
	Keywords	_asm, _endasm, static
	Command-line options	-Opa+, -Opa-
	Bit values	0, 1
	Constants	0xFF, 'A'
Italic Courier New	A variable argument	<i>file.o</i> , where <i>file</i> can be any valid filename
Square brackets []	Optional arguments	mcc18 [options] <i>file</i> [options]
Curly brackets and pipe character: { }	Choice of mutually exclusive arguments; an OR selection	errorlevel {0 1}
Ellipses...	Replaces repeated text	var_name [, var_name...]
	Represents code supplied by user	void main (void) { ... }

WARRANTY REGISTRATION

Please complete the enclosed Warranty Registration Card and mail it promptly. Sending in the Warranty Registration Card entitles users to receive new product updates. Interim software releases are available at the Microchip web site.

RECOMMENDED READING

This user's guide describes how to use the PICkit™ Serial Analyzer. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

Readme Files

For the latest information on using other tools, read the tool-specific Readme files in the Readmes subdirectory of the MPLAB IDE installation directory. The Readme files contain update information and known issues that may not be included in this user's guide.

THE MICROCHIP WEB SITE

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers and other language tools. These include the MPLAB® C18 and MPLAB C30 C compilers; MPASM™ and MPLAB ASM30 assemblers; MPLINK™ and MPLAB LINK30 object linkers; and MPLIB™ and MPLAB LIB30 object librarians.
- **Emulators** – The latest information on Microchip in-circuit emulators. This includes the MPLAB ICE 2000 and MPLAB ICE 4000.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debugger, MPLAB ICD 2.
- **MPLAB® IDE** – The latest information on Microchip MPLAB IDE, the Windows® Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB SIM simulator, MPLAB IDE Project Manager

PICKit™ Serial Analyzer User's Guide

and general editing and debugging features.

- **Programmers** – The latest information on Microchip programmers. These include the MPLAB PM3 and PRO MATE® II device programmers and the PICSTART® Plus and PICKit™ 2 development programmers.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: <http://support.microchip.com>

DOCUMENT REVISION HISTORY

Revision A (January 2007)

- Initial release of this document.

Chapter 1. PICKit™ Serial Analyzer Overview

1.1 INTRODUCTION

The PICKit™ Serial Analyzer development system enables a personal computer (PC) to communicate with embedded development systems via serial protocols such as I²C™, SPI, asynchronous and synchronous USART. The PC program uses a graphical interface to enter data and commands to communicate to the target device. Data and commands can be entered using basic or scripting commands. The PICKit™ Serial Analyzer connects to the embedded development system using a 6-pin header.

The PICKit™ Serial Analyzer is a sophisticated and highly configurable device. Please take a few moments to familiarize yourself with the hardware interface and PC program by reading this user's guide. **Chapter 2. "Getting Started"** will guide you through installing the PC program and running a simple demonstration program on the 28-Pin Demo Board (DM164120-3) using the I²C serial protocol.

1.2 HIGHLIGHTS

This chapter discusses:

- PICKit™ Serial Analyzer Contents
- PICKit™ Serial Analyzer Development System
- PICKit™ Serial Analyzer Hardware
- PICKit™ Serial Analyzer PC Software

1.3 PICKit™ SERIAL ANALYZER CONTENTS

The PICKit™ Serial Analyzer serial communications development system contains the following items:

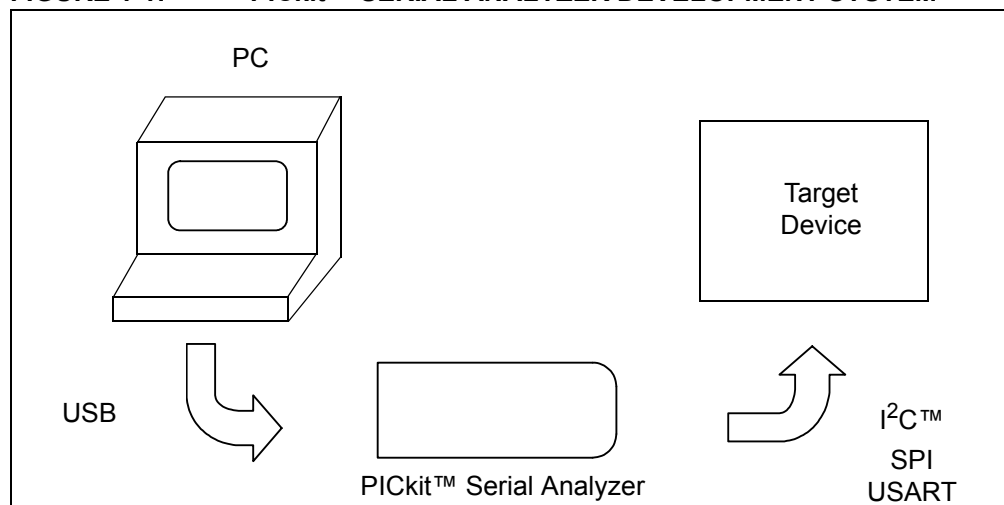
1. The PICKit™ Serial Analyzer
2. USB cable
3. PICKit™ Serial Analyzer CD-ROM

1.4 PICKit™ SERIAL ANALYZER DEVELOPMENT SYSTEM

The PICKit™ Serial Analyzer consists of several components that together make an embedded serial communications development system. The PC program runs on Microsoft® Windows® compatible computers with a USB port. The PICKit™ Serial Analyzer connects to the PC using a USB cable. Finally, the PICKit™ Serial Analyzer interfaces to the target device using a 6-pin header. Figure 1-1 illustrates the PICKit™ Serial Analyzer embedded serial communications development system.

PICKit™ Serial Analyzer User's Guide

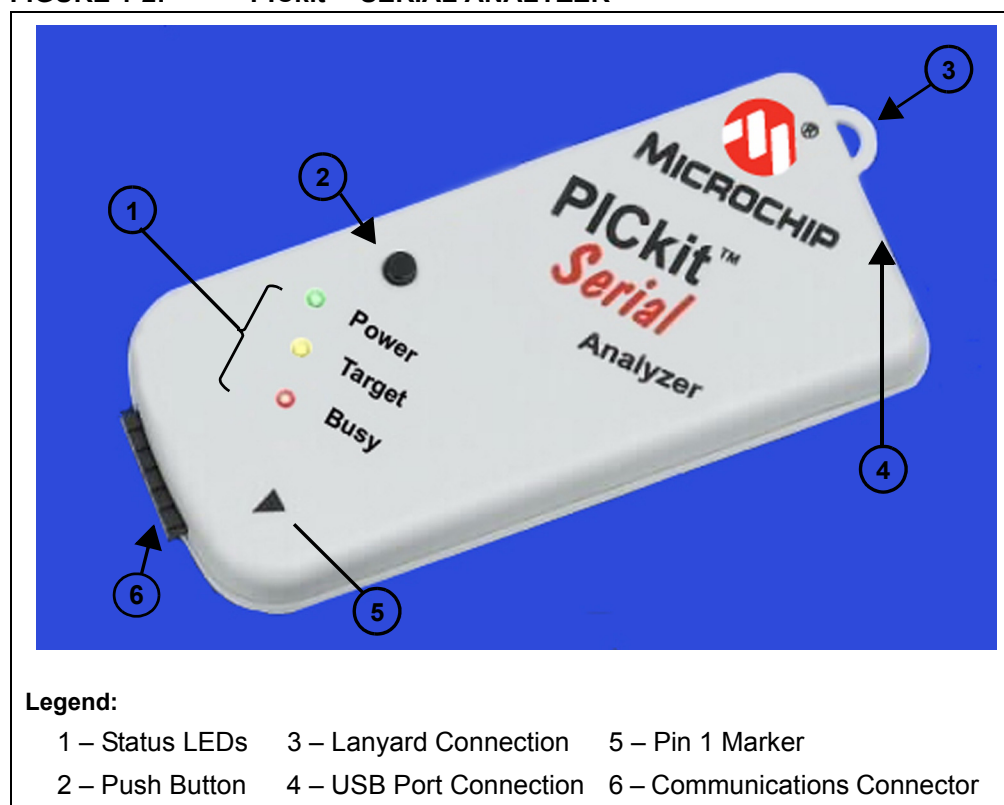
FIGURE 1-1: PICKit™ SERIAL ANALYZER DEVELOPMENT SYSTEM



1.5 PICKit™ SERIAL ANALYZER HARDWARE

The PICKit™ Serial Analyzer connects to a Microsoft® Windows® compatible computer using a USB port. It interfaces to the target device using a 6-pin header. Figure 1-2 shows an overview of the PICKit™ Serial Analyzer.

FIGURE 1-2: PICKit™ SERIAL ANALYZER



PICKit™ Serial Analyzer Overview

1.5.1 Status LEDs

The Status LEDs indicate the status of the PICKit™ Serial Analyzer.

1. Power (green) – Power is applied to the PICKit™ Serial Analyzer by the USB port.
2. Target (yellow) – The PICKit™ Serial Analyzer is communicating with the target device.
3. Busy (red) – The PICKit™ Serial Analyzer is communicating with the target device.

1.5.2 Push Button

The push button is available for future implementation.

1.5.3 Lanyard Connection

To help prevent possible loss of the PICKit™ Serial Analyzer, a convenient lanyard connection is available.

1.5.4 USB Port Connection

The USB Port Connection is a USB mini-B connector. Connect the PICKit™ Serial Analyzer to the PC using the supplied cable.

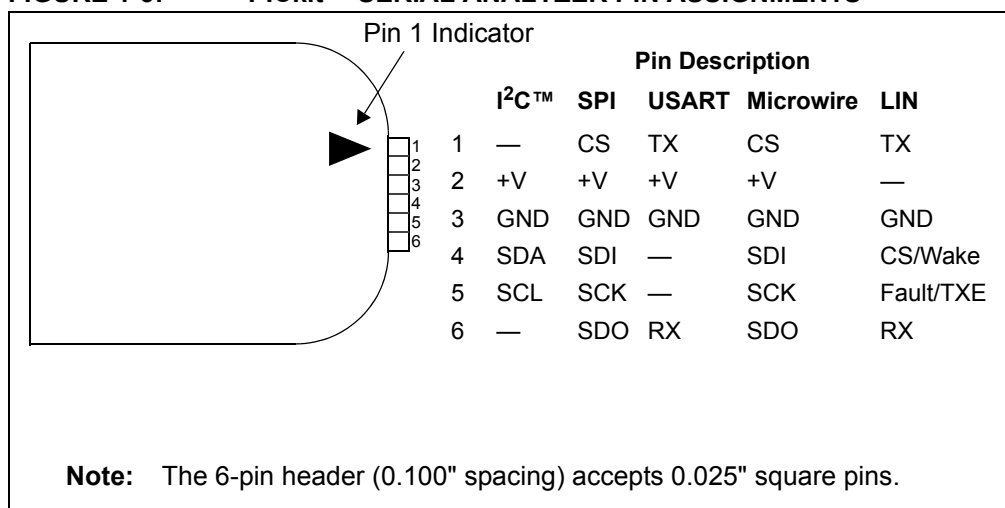
1.5.5 Pin 1 Marker

The Pin 1 marker assists in aligning the PICKit™ Serial Analyzer with the target device. Pin assignments are shown in Figure 1-3.

1.5.6 Communication Connector

The communication connector connects to the target device using an inexpensive 6-pin, 0.100" pitch spacing, 0.025" square pin header. Pin assignments are shown in Figure 1-3.

FIGURE 1-3: PICKit™ SERIAL ANALYZER PIN ASSIGNMENTS



1.6 PICKit™ SERIAL ANALYZER SOFTWARE

1.6.1 PC Program

The PICKit™ Serial Analyzer PC program uses a graphical interface to enter data and commands to communicate to the target device. Data and commands can be entered using basic or scripting commands. **Chapter 3. “PICKit™ Serial Analyzer PC Program”** explains the installation and operation of the program. Following Chapter 3 there are individual chapters that explain the specific serial communications modes and their operation.

1.6.2 Dynamically Linked Library (DLL)

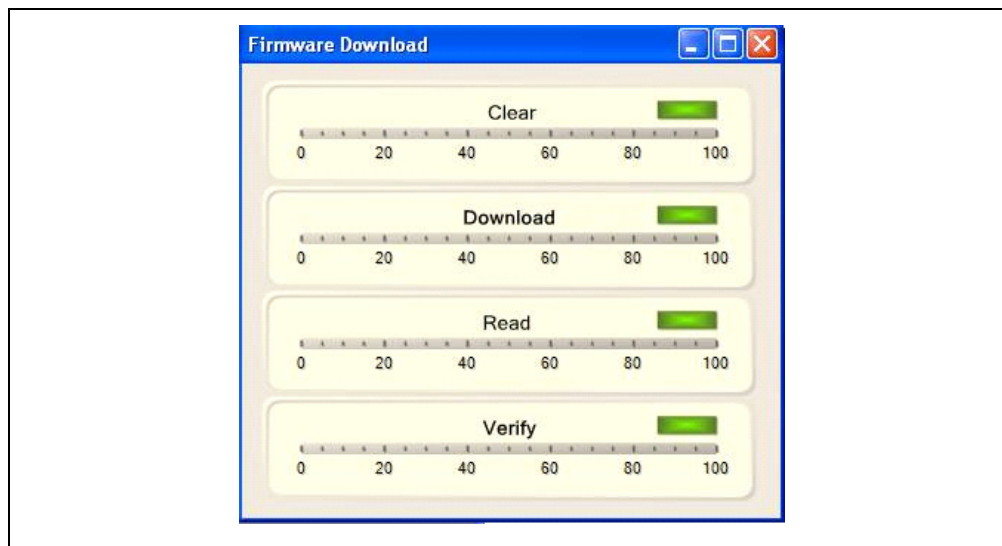
The PICKit™ Serial Analyzer DLL is explained in **Chapter 12. “PICKit™ Serial Analyzer DLL”**.

1.6.3 Firmware

The PICKit™ Serial Analyzer firmware is explained in **Chapter 11. “PICKit™ Serial Analyzer Firmware”**.

The latest version of the PICKit™ Serial Analyzer firmware can be downloaded from the Microchip Technology web site. The firmware is updated by selecting *PICKit Serial Analyzer > Download PICKit Serial Analyzer Firmware* from the menu bar. An open file window will open. Select the *.hex file to be uploaded to the PICKit™ Serial Analyzer and click on the **Open** button. The Firmware Download window will open as shown in Figure 1-4 to indicate the status of the firmware update.

FIGURE 1-4: FIRMWARE DOWNLOAD WINDOW



Chapter 2. Getting Started

2.1 INTRODUCTION

This chapter will get you started using the PICKit™ Serial Analyzer with the 28-Pin Demo Board. In this demo, the PICKit™ Serial Analyzer will communicate with the 28-Pin Demo Board using the I²C serial protocol. The PICKit™ Serial Analyzer will be the I²C Master and the 28-Pin Demo Board will be the I²C Slave device. The 28-Pin Demo board is programmed to emulate an I²C real-time clock and Serial EEPROM. For more information about the 28-Pin Demo Board hardware, see the **28-Pin Demo Board User's Guide** (DS41301).

For more information about the 28-Pin Demo Board I²C™ demo firmware, see **Appendix B. "28-Pin Demo Board I²C™ Demonstration Firmware"**.

The demo program source code and *.hex file can be found on the PICKit™ Serial CD-ROM at D:\28-pin Demo Board\Firmware\.

2.2 HIGHLIGHTS

This chapter discusses:

- Installing the PICKit™ Serial Analyzer Software
- Connecting the PICKit Serial Analyzer to the PC
- Connecting the PICKit Serial Analyzer to the 28-Pin Demo Board
- Starting the PICKit Serial Analyzer Program
- Running The 28-Pin Demo I²C™ Demonstration Program
- I²C Communications – Basic Operations
- 28-Pin Demo I²C™ Source Code and Firmware

2.3 INSTALLING THE PICKit™ SERIAL ANALYZER SOFTWARE

Insert the PICKit™ Serial Analyzer CD-ROM into the CD-ROM drive. In a few moments the introductory screen should be displayed. Follow the directions on the screen to install the PICKit Serial Analyzer software.

If the introductory screen does not appear, browse to the CD-ROM directory and select the AutorunPro.exe program.

Note: The PICKit™ Serial Analyzer program requires the Microsoft® .NET Framework Version 2.0. If the .NET Framework is not installed on your computer (or if in doubt), select the application plus Microsoft® .NET Framework installation.

2.4 CONNECTING THE PICKit™ SERIAL ANALYZER TO THE PC

Connect the PICKit Serial Analyzer to the PC using the supplied USB cable. There are no USB drivers to install. The green Power indicator should light indicating that the PICKit Serial Analyzer is powered.

PICKit™ Serial Analyzer User's Guide

2.5 CONNECTING THE PICKit™ SERIAL ANALYZER TO THE 28-PIN DEMO BOARD

Connect the PICKit Serial Analyzer to P3 on the 28-Pin Demo Board as shown in Figure 2-1. The PICKit Serial Analyzer will supply power to the 28-Pin Demo Board and perform a power on routine:

- LEDs will flash in sequence DS1, DS2, DS3, DS4, DS3, DS2, and DS1 twice
- All LEDs will turn off
- All LEDs will turn on
- All LEDs will turn off
- LEDs will display in hexadecimal: A, D, C
- LEDs will display the top 4 bits of the ADC value read from potentiometer RP1

FIGURE 2-1: CONNECTING PICKit™ SERIAL TO THE 28-PIN DEMO BOARD



2.6 STARTING THE PICKit™ SERIAL ANALYZER PROGRAM

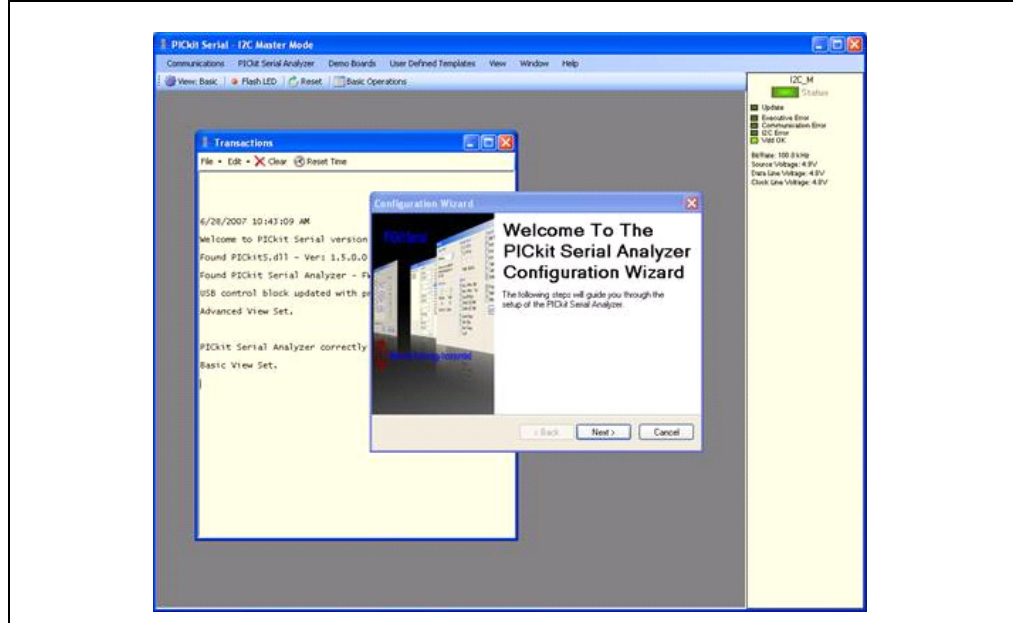
You can start the program by:

- Clicking on the desktop icon, or
- Navigating to *Start>All Programs>Microchip>PICKit Serial Analyzer*

After a few moments, the program will start and display the main window as shown in Figure 2-2.

If this is the first time you are running the program, the Configuration Wizard will automatically run. Click on the **Next** button and accept the default settings for I²C Master mode. For more information about using the I²C Master mode, see **Chapter 4. "I²C™ Master Communications."**

FIGURE 2-2: PICKIT™ SERIAL ANALYZER MAIN WINDOW



2.7 RUNNING THE 28-PIN DEMO I²C™ DEMONSTRATION PROGRAM

Select the 28-Pin Demo I²C demonstration by clicking on *Demo Boards > 28 Pin Demo I²C* from the menu bar. The 28-Pin Demo I²C demonstration window will be displayed as shown in Figure 2-3.

The Real-Time Clock (RTC) will be displayed first. Note the tabs to select between the RTC, EEPROM and ADC demonstrations. The demonstration program will constantly poll the 28-Pin Demo Board and display the contents of the real-time clock and the ADC.

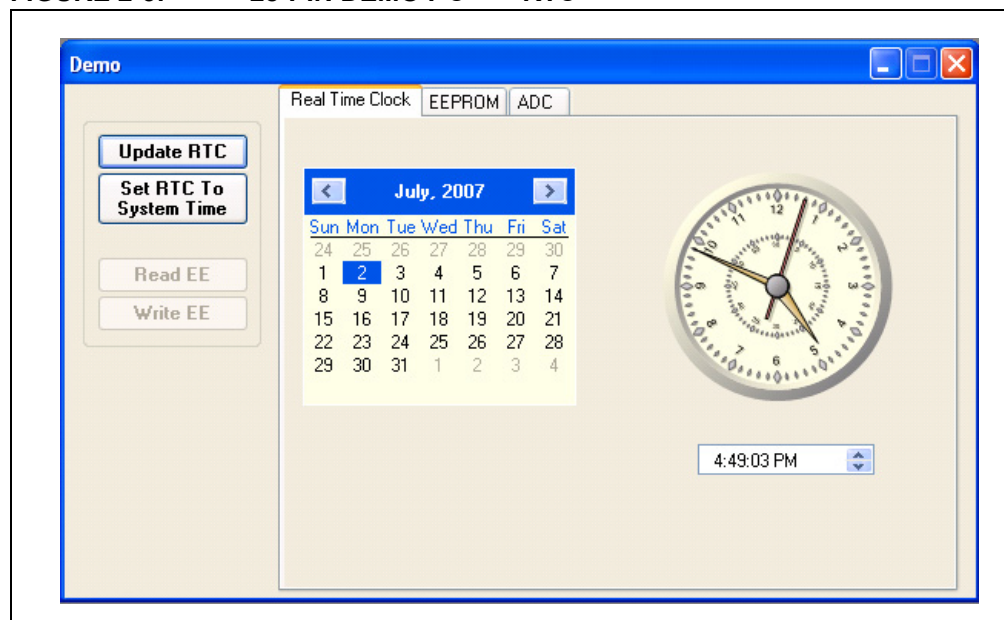
2.7.1 Real-Time Clock (RTC)

Clicking on the **Real-Time Clock** tab will display calendar and clock contents of the real-time clock function running on the 28-Pin Demo Board. The 28-Pin Demo Board has been programmed to emulate a stand-alone serial I²C clock-calendar device. The I²C commands are very similar to the commands used in these devices. The demonstration program will constantly poll the 28-Pin Demo Board and display the contents of the real-time clock.

The Real-Time Clock window displays calendar and clock controls. Notice the date and time when the 28-Pin Demo Board has first been powered on. The date and time start at January 1, 2000 at midnight (12:00 AM).

The user can manually enter calendar and clock values and send the values to the real-time clock by clicking on the **Update RTC** button. Or the user can click on the **Set RTC to System Time** button to set the real-time clock to the date and time of the computer.

FIGURE 2-3: 28-PIN DEMO I²C™ – RTC



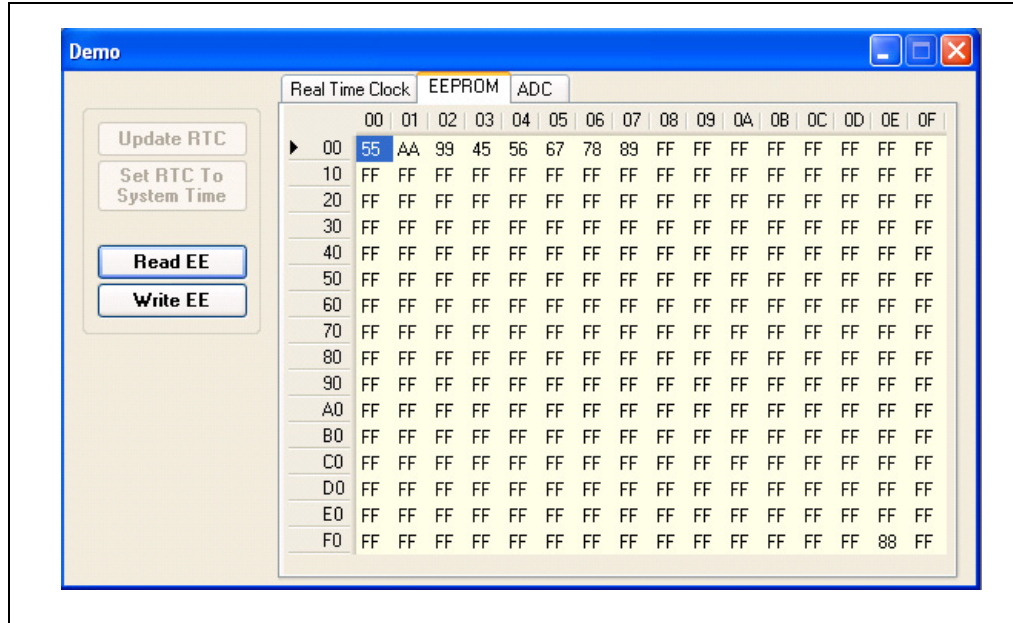
2.7.2 Serial EEPROM (EEPROM)

Clicking on the EEPROM tab will display the 256 byte array of EEPROM memory as shown in Figure 2-4. The 28-Pin Demo Board has been programmed to emulate a stand-alone serial I²C EEPROM device such as a 24LC02. The I²C commands are very similar to the commands used in these devices.

The Serial EEPROM tab displays the contents of a serial EEPROM implemented on the 28-Pin Demo Board. When this tab is first displayed, the values are grayed out. This means that the display does not match the contents of the emulated serial EEPROM. Click on **Read EE** button and the program will read and display the contents of the 28-Pin Demo Board. Notice that the displayed values are now black.

Individual memory locations can be changed by clicking on the value and typing in a new value in hexadecimal. Notice that the changed values will be displayed in red. This means the value has changed but has not been written to the emulated serial EEPROM. Click on the **Write EE** button and the values will be written. The color of the value will turn to black indicating that the value has been written and the display matches the contents of the emulated serial EEPROM.

FIGURE 2-4: 28-PIN DEMO I²C™ – EEPROM

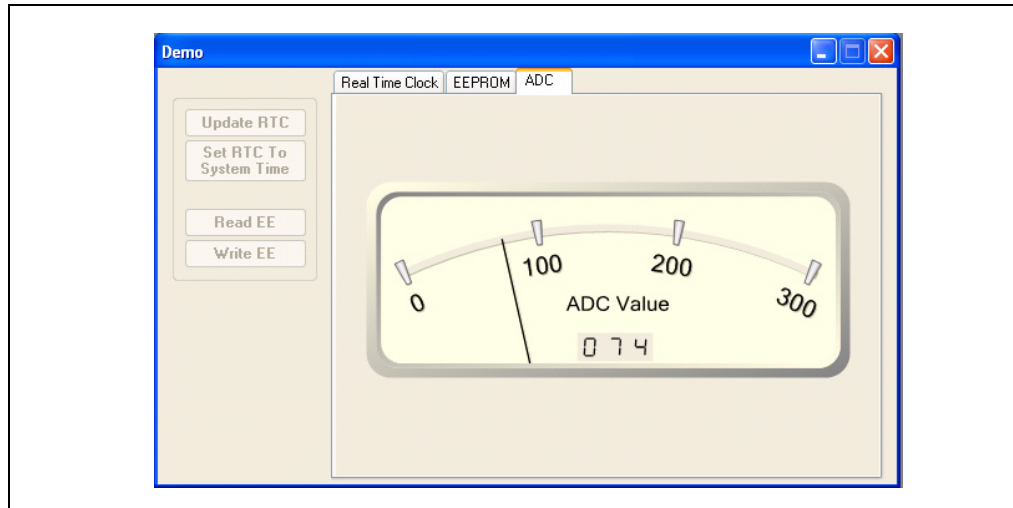


2.7.3 Analog-to-Digital Converter (ADC)

Clicking on the ADC tab will show a meter gauge displaying the value of the ADC as read from potentiometer RP1 as shown in Figure 2-5.

The meter gauge displays the Most Significant 8 bits of the 10-bit ADC internal to the PIC[®] microcontroller. Rotate potentiometer RP1 and the display changes almost instantaneously. The demonstration program will constantly poll the 28-Pin Demo Board and display the contents of the ADC.

FIGURE 2-5: 28-PIN DEMO I²C™ – ADC



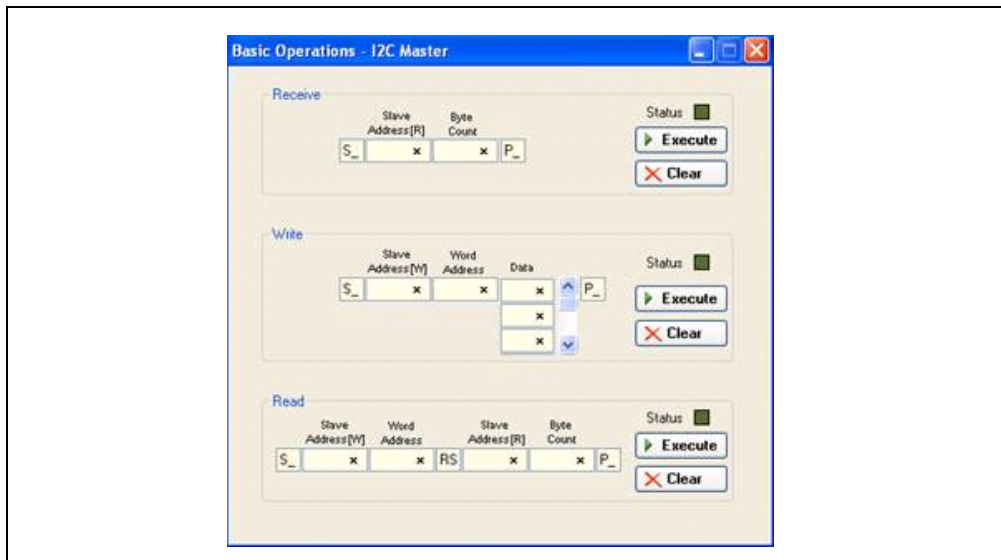
PICKit™ Serial Analyzer User's Guide

2.8 I²C™ COMMUNICATIONS – BASIC OPERATIONS

Individual I²C commands and data can be read and written to the 28-Pin Demo Board from the Basic Operations window as shown in Figure 2-6. Ensure that the PICKit Serial Analyzer program is in I²C Master mode by selecting *PICKit Serial Analyzer > Run Configuration Wizard* from the menu bar and selecting I²C Master.

Note: The 28-Pin Demo I²C window and the Basic Operations window cannot be opened at the same time. When the 28-Pin Demo I²C window is opened, the Basic Operations window will automatically close.

FIGURE 2-6: I²C™ BASIC OPERATIONS



2.8.1 Real-Time Clock (RTC)

The Slave address for the emulated real-time clock on the 28-Pin Demo Board is hexadecimal A2 (0xA2). The Word Address selects the following memory locations:

TABLE 2-1: MEMORY LOCATIONS

Word Address	Contents
0x00	Configuration 1
0x01	Configuration 2
0x02	Seconds
0x03	Minutes
0x04	Hours
0x05	Days
0x06	Weekdays
0x07	Months
0x08	Years

For example, to read seconds from the real-time clock:

Step 1 – Enter 0xA2 into the Slave Address[W] block in the Read section of the Basic Operations window (lower third of window)

Step 2 – Enter 0x02 into the Word Address block

Step 3 – Note that the Slave Address[R] has already been entered for you (the Read bit is set).

Step 4 – Enter 0x01 into the Byte Count block

Step 5 – Click on the **Execute** button

The I²C combination command (Write then Read) will be sent to the 28-Pin Demo Board. The command and the contents of Word Address 0x02 (seconds) will be displayed in the transaction window as shown in Figure 2-7.

FIGURE 2-7: RTC TRANSACTIONS DEMO



2.8.2 EEPROM

The Slave address for the emulated Serial EEPROM on the 28-Pin Demo Board is hexadecimal A8 (0xA8). The Word Address selects one of 256 8-bit memory locations:

TABLE 2-2: WORD ADDRESS CONTENTS

Word Address	Contents
0x00	Memory Contents
...	...
0xFF	Memory Contents

2.8.3 ADC

The Slave address for the ADC on the 28-Pin Demo Board is hexadecimal AA (0xAA). The Word Address 0x01 selects the memory location containing the Most Significant 8 bits of the 10-bit ADC of the PIC microcontroller.

2.9 28-PIN DEMO I²C™ SOURCE CODE AND FIRMWARE

The demo program source code and *.hex file can be found on the PICkit Serial CD-ROM at *D:\28-pin Demo Board\Firmware*.

PICKit™ Serial Analyzer User's Guide

NOTES:

Chapter 3. PICKit™ Serial Analyzer PC Program

3.1 INTRODUCTION

This chapter covers the installation, starting and high level operations of the PICKit Serial Analyzer program. Detailed information about the entering of data and commands for specific serial communications modes are given in the following chapters.

3.2 HIGHLIGHTS

This chapter discusses:

- Installing The PICKit Serial Analyzer Software
- Starting the Program
- Configuration Wizard
- Main Window
- Specific Communications Modes

3.3 INSTALLING THE PICKit™ SERIAL ANALYZER SOFTWARE

Insert the PICKit Serial Analyzer CD-ROM into the CD-ROM drive. In a few moments the introductory screen should be displayed. Follow the directions on the screen to install the PICKit Serial Analyzer software.

If the introductory screen does not appear, browse to the CD-ROM directory and select the AutorunPro.exe program.

Note: The PICKit Serial Analyzer program requires the Microsoft® .NET Framework Version 2.0.

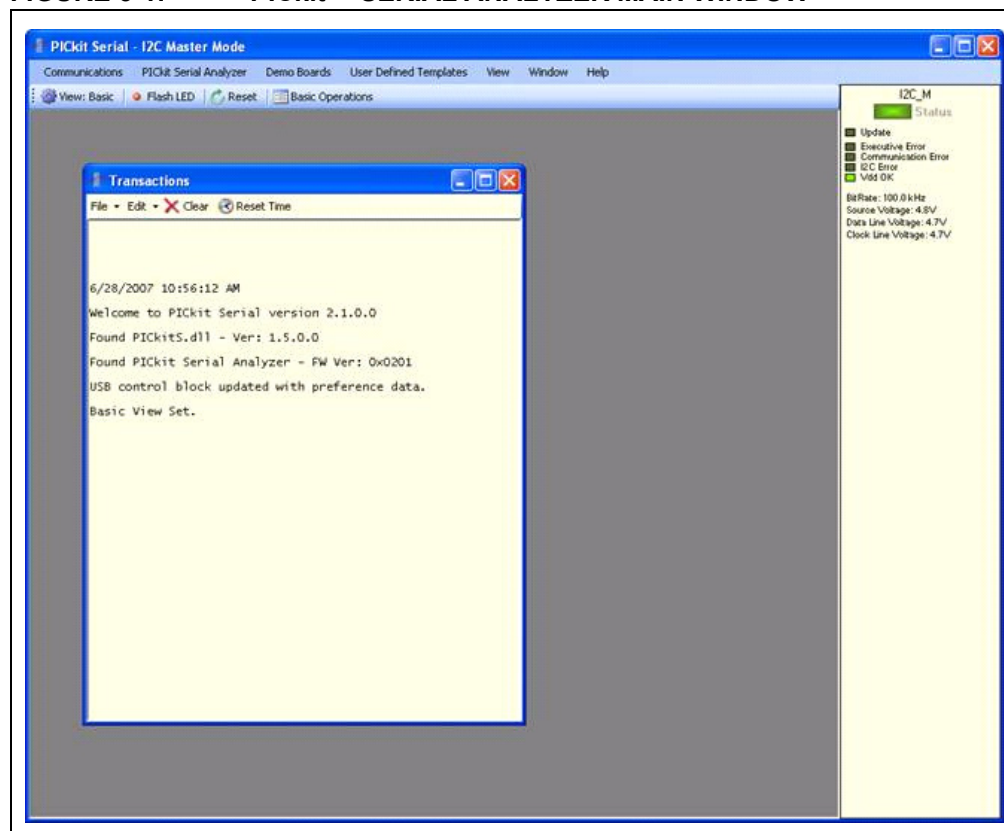
3.4 STARTING THE PROGRAM

You can start the program by

- Clicking on the desktop icon, or
- Navigating to *Start>All Programs>Microchip>PICKit Serial Analyzer*

After a few moments, the program will start and display the main window as shown in Figure 3-1.

FIGURE 3-1: PICKit™ SERIAL ANALYZER MAIN WINDOW



3.5 CONFIGURATION WIZARD

If it is the first time that the PICKit Serial Analyzer program is run, the Configuration Wizard will run automatically. The Configuration Wizard can be manually invoked by selecting *PICKit Serial Analyzer > Run Configuration Wizard* from the menu bar.

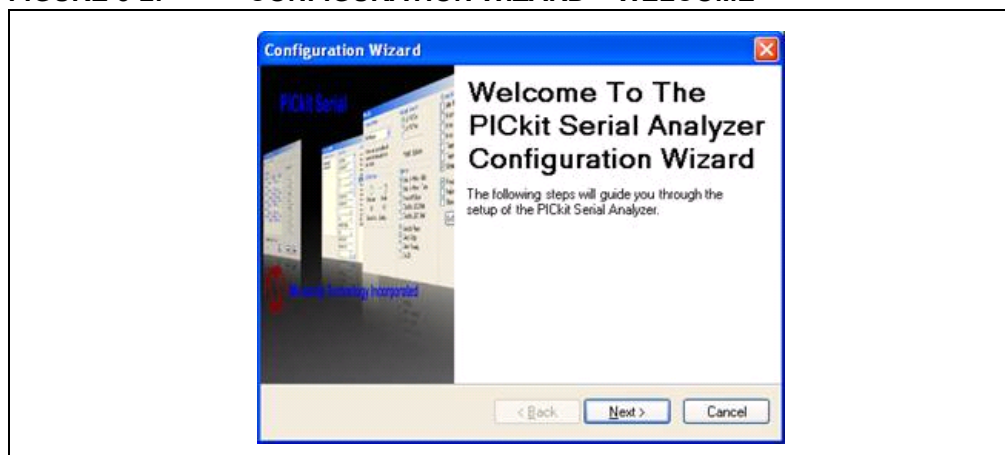
The Configuration Wizard will guide you through the basic steps to configure the PICKit Serial Analyzer program for a specific communications mode (I²C Master, I²C Slave, SPI, USART, LIN, Microwire).

Advanced configuration can be done from the Configuration Window by selecting *PICKit Serial Analyzer > Configure Communications Mode* from the menu bar.

As an example, Figure 3-2 through Figure 3-7 show how to configure for I²C Master mode. Refer to the specific communications chapter for detailed information on the Configuration Wizard for that communications mode.

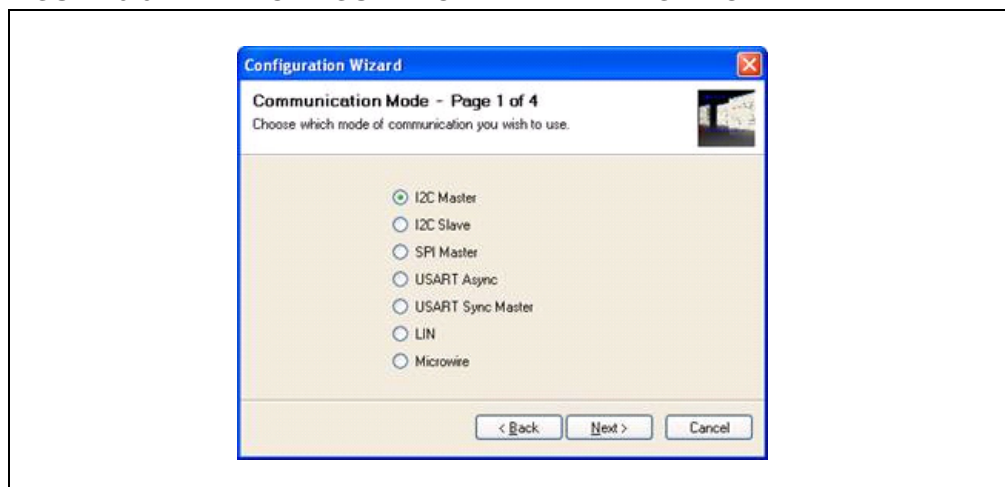
The Configuration Wizard Welcome window is shown in Figure 3-2. You may choose to continue by clicking on the **Next** button or canceling the wizard by clicking on the **Cancel** button.

FIGURE 3-2: CONFIGURATION WIZARD – WELCOME



The Configuration Wizard Page 1 of 4, as shown in Figure 3-3, displays the available communications modes and allows you to choose one of the modes.

FIGURE 3-3: CONFIGURATION WIZARD – PAGE 1 OF 4



In this example, I²C Master Communication's mode is selected. The Configuration Wizard Page 2 of 4, as shown in Figure 3-4, allows you to select the bus speed. A more comprehensive list of bus speeds can be chosen from the Configuration Window by selecting *PICKit Serial Analyzer > Configure Communications Mode* from the menu bar.

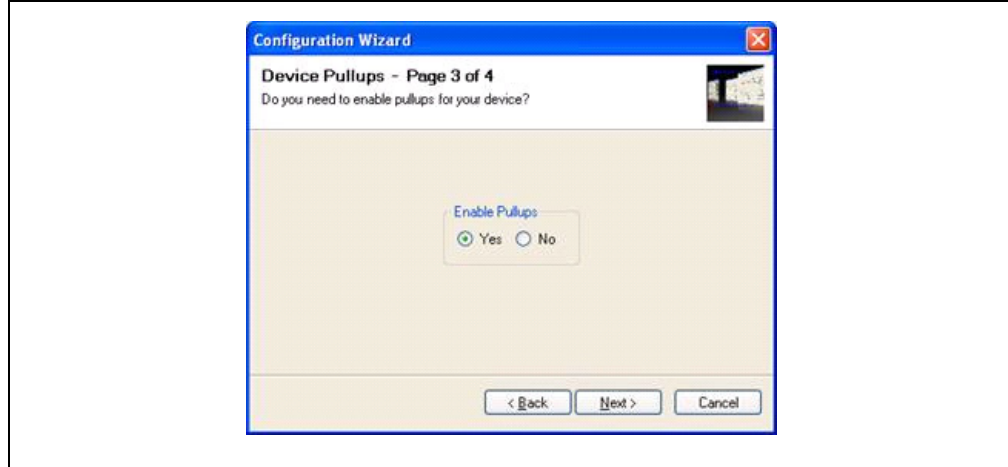
FIGURE 3-4: CONFIGURATION WIZARD – PAGE 2 OF 4



PICKit™ Serial Analyzer User's Guide

The I²C bus requires pull-up resistors. The PICKit Serial Analyzer has the ability to enable internal 2.2 kΩ pull-up resistors. If the target device does not have pull-up resistors installed, then enable pull-ups by selecting the **Yes** radio button as shown in Figure 3-5. If the target device has the pull-up resistors installed, you can disable the internal pull-ups by selecting the **No** radio button.

FIGURE 3-5: CONFIGURATION WIZARD – PAGE 3 OF 4



The PICKit Serial Analyzer can power the target device from 0 to 5 VDC at a combined total current limit of 100 mA (PICKit Serial Analyzer plus target device). The Configuration Wizard Page 4 of 4, as shown in Figure 3-6, allows you to choose between powering the target device and selecting the specific target voltage.

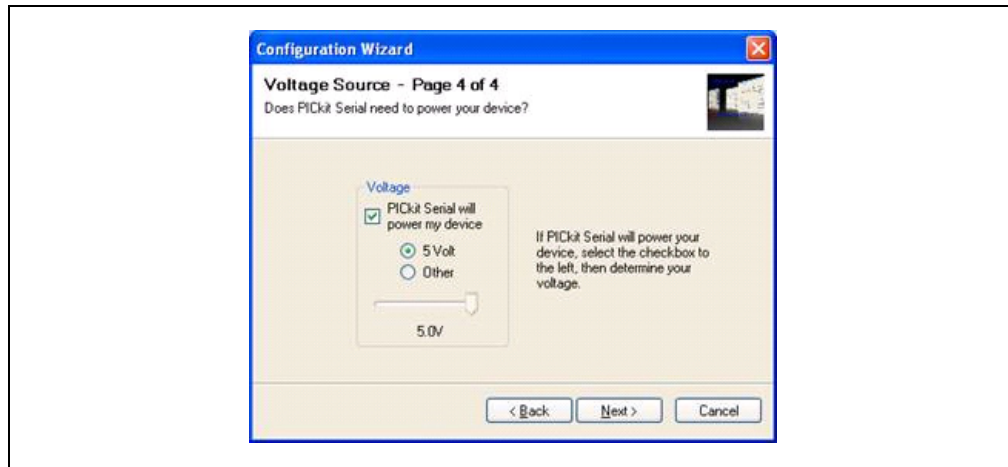
CAUTION

Even though the voltage can be set as low as 0 VDC, it is up to the user to verify the required operating voltage of the target device.

CAUTION

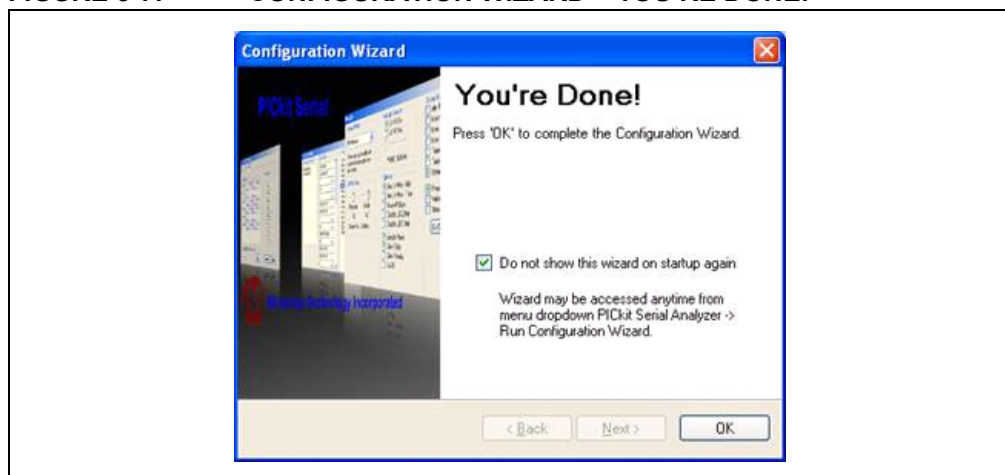
The USB port current limit is set to 100 mA. If the target plus PICKit Serial Analyzer exceeds this current limit, the USB port will turn off. The target may be powered externally if more power is required.

FIGURE 3-6: CONFIGURATION WIZARD – PAGE 4 OF 4



Once all pages of the Configuration Wizard are completed, you can choose to not display the wizard at start up by checking the Do not show this wizard on start-up again check box.

FIGURE 3-7: CONFIGURATION WIZARD – YOU'RE DONE!

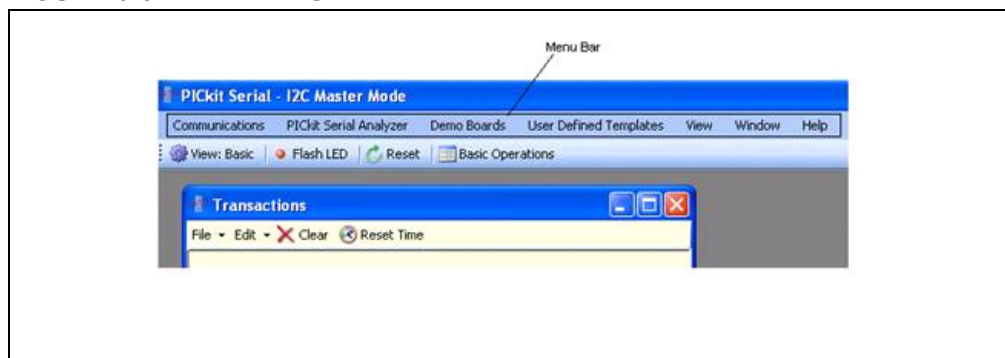


3.6 MAIN WINDOW

3.6.1 Menu Bar

The menu bar selects various functions of the PICKit Serial Analyzer program. A summary of the functions are:

FIGURE 3-8: MENU BAR



COMMUNICATIONS

The Communications menu selections display operation windows to enter data and commands to communicate with the target device.

- Basic Operations – Displays the Basic Operations window for the communications mode selected (see PICKit Serial Analyzer -> Select Communications Mode)
- Script >
 - Script Builder – Displays the Script Builder window
 - Script Execute – Displays the Script Execute window
- I²C Slave Profile – Displays the I²C Slave Profile Generator (Enabled only in I²C Slave mode)

PICKit™ Serial Analyzer User's Guide

PICKit™ SERIAL ANALYZER

The PICKit Serial Analyzer menu selection commands the PICKit Serial Analyzer hardware.

- Select Communications Mode >
 - I²C Master – Puts the PICKit Serial Analyzer in I²C Master Communications mode
 - SPI Master – Puts the PICKit Serial Analyzer in SPI Master Communications mode
 - USART Asynchronous – Puts the PICKit Serial Analyzer in USART Asynchronous Communications mode
 - USART Synchronous Master – Puts the PICKit Serial Analyzer in USART Synchronous Master Communications mode
 - I²C Slave – Puts the PICKit Serial Analyzer in I2C Slave Communications mode
 - LIN – Puts the PICKit Serial Analyzer in LIN Communications mode
 - Microwire – Puts the PICKit Serial Analyzer in Microwire Communications mode
- Configure Communications Mode – Displays the Configuration Communications Mode window for the communications mode selected (see PICKit Serial Analyzer -> Select Communications Mode)
- Download PICKit Serial Analyzer Firmware – Displays the Firmware Download window. Firmware updates are available from the Microchip Technology web site.
- Run Configuration Wizard – Displays the Configuration Wizard
- Perform System Reset – Closes and then reinitializes USB communications to the PICKit Serial Analyzer
- Reset PICKit Serial Analyzer – Resets the PICKit Serial Analyzer if an error condition is present
- PICKit Serial Analyzer No. – Up to four PICKit Serial Analyzers can be controlled from the PC software. The number is assigned to the hardware as it enumerates on the USB bus.

DEMO BOARDS

The Demo Boards menu selection displays the selected demonstration window. The PICKit Serial Analyzer program will be automatically configured for the communications mode of the selected demonstration.

- 28-Pin Demo I²C – Displays the 28-Pin Demo Board I²C demo graphical user interface. For more information see **Appendix B**. “28-Pin Demo Board I²C™ Firmware.”

USER DEFINED TEMPLATES

- Display Template – Displays the User Defined Template window
- My Templates – Selects and displays template windows created by the user

VIEW

- Basic – The PICKit Serial Analyzer program will display basic commands and status view
- Advanced – The PICKit Serial Analyzer program will display advanced commands and status view

WINDOW

- New Transaction Window – Opens new or additional transaction window. Multiple

transaction windows can be opened as needed for logging communications.

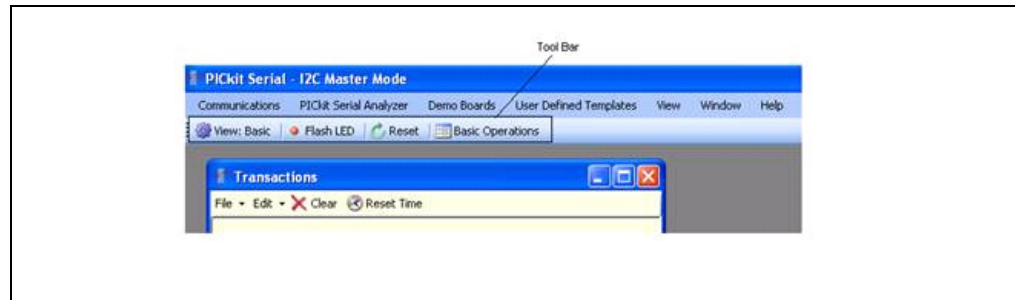
- Close All – Closes all windows
- Cascade – Cascade windows
- Tile Horizontally – Tile windows horizontally
- Tile Vertically – Tile windows vertically

HELP

- About – Displays program version information
- Show PICKit Serial Analyzer Connections – Displays pinout for current communications mode
- Show Event Bytes – Displays Event Marker code for current communication mode
- User's Guide – Displays the PICKit Serial User's Guide

3.6.2 Tool Bar

FIGURE 3-9: TOOL BAR



The Tool Bar gives quick access to often used commands. These commands are also available from the Menu Bar.

- View: Basic/Advanced – toggles between Basic and Advanced views
- Flash LED – Flashes the Red Busy LED to confirm communications
- Reset – Resets the PICKit Serial Analyzer if an error condition is present
- Basic Operations – Displays the Basic Operations window for the communications mode selected
- Configure – (Advanced View Only) – Displays the configuration window for the communications mode selected

3.6.3 Status Column

The Status Column displays status information for the selected serial communications mode. In Basic View mode, a simplified status is displayed as shown in Figure 3-10. In Advanced View mode additional status information is displayed for the communications mode selected as shown in Figure 3-11.

The status information that is displayed depends on the selected communications mode (I²C, SPI, USART, etc.). The following chapters give more detailed explanation of the status window for the particular serial communications mode.

FIGURE 3-10: STATUS COLUMN (BASIC VIEW)

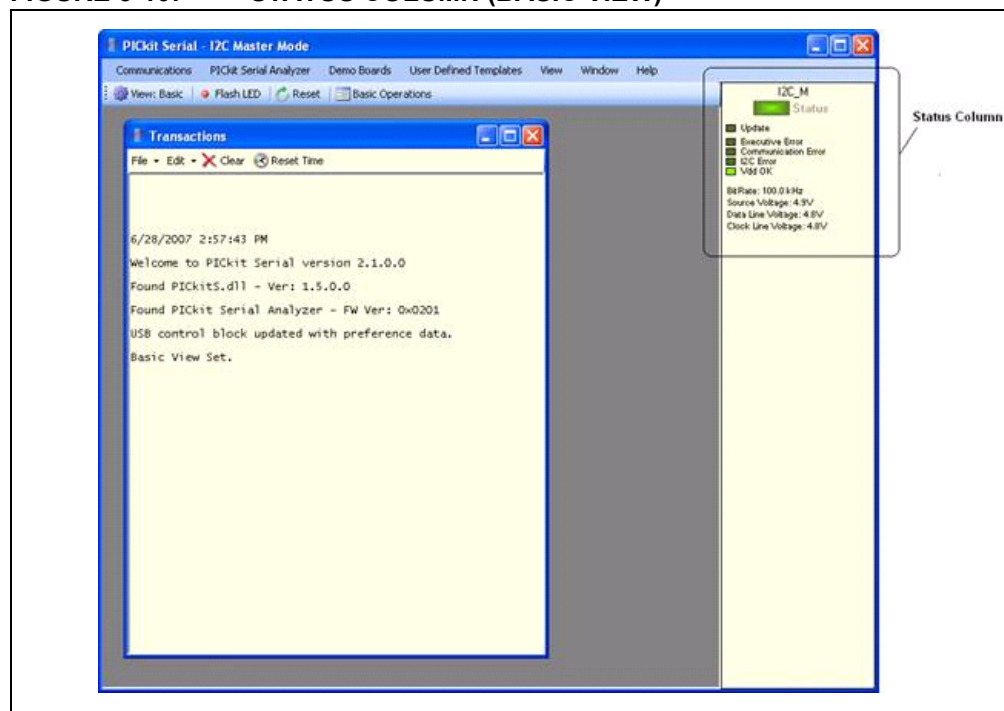
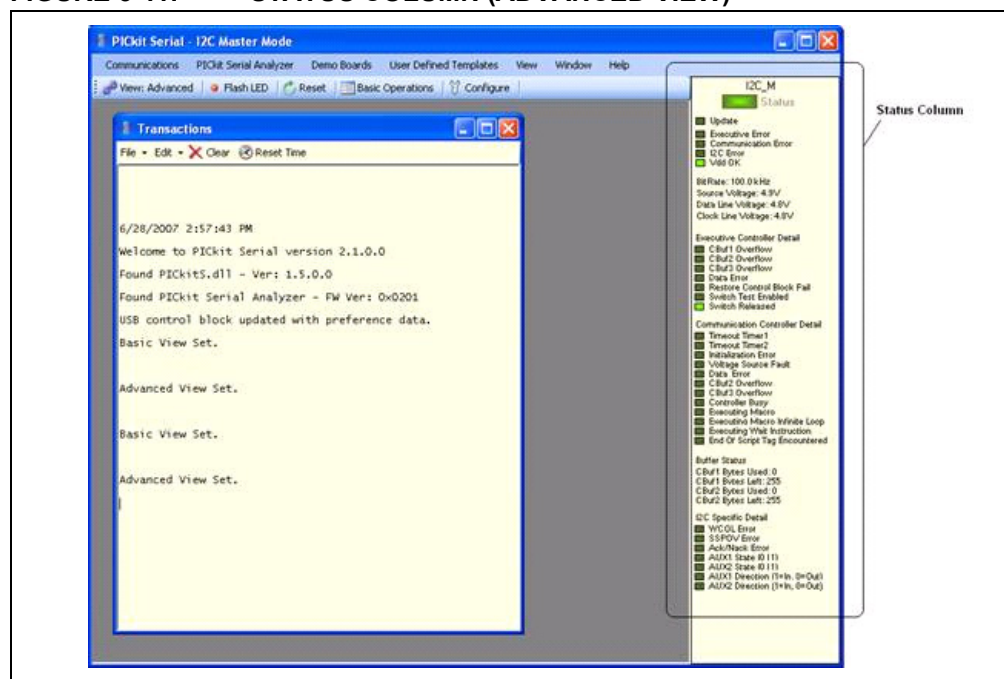


FIGURE 3-11: STATUS COLUMN (ADVANCED VIEW)



3.6.4 Transactions Window

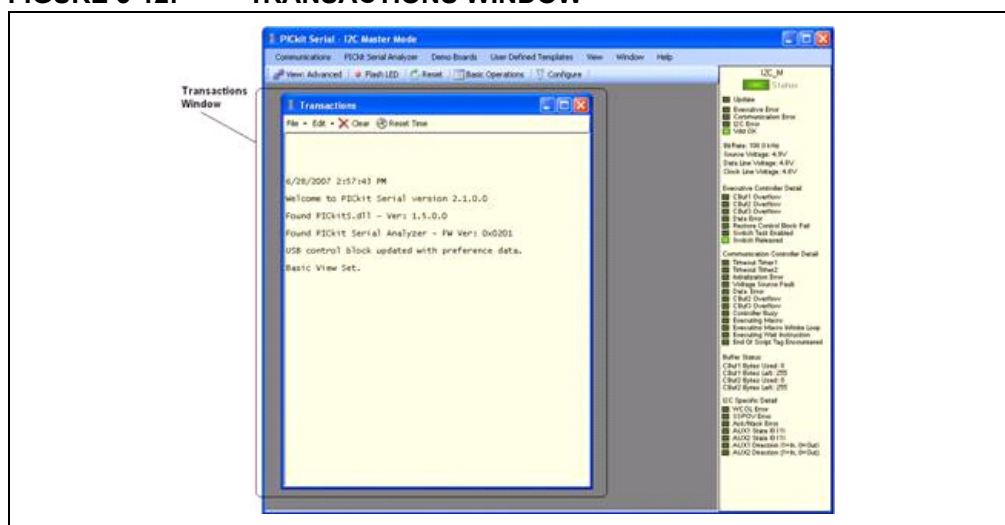
The Transactions window, shown in Figure 3-12, keeps a running log of the commands and data that are communicated between the PICKit Serial Analyzer program and target device.

From the menu bar on the Transaction window, the contents can be saved (*File>Save*) to a *.txt or *.rtf file. The file can later be retrieved (*File>Open*) and displayed in the Transactions window.

PICKit™ Serial Analyzer PC Program

Additional Transactions windows can be displayed. From the PICKit Serial Analyzer menu bar, select *Window > New Transaction Window*. The active Transactions window will log the current commands and data.

FIGURE 3-12: TRANSACTIONS WINDOW



FILE

- **Open** – Opens a *.txt or *.rtf file and displays it in the Transactions window
- **Save** – Saves the contents of the Transactions window to a *.txt or *.rtf file
- **Close** – Closes the selected Transactions window

EDIT

- **Copy** – The selected contents of the Transactions window will be copied to the clipboard
- **Paste** – The contents of the clipboard will be pasted into the Transactions window
- **Select All** – contents of the Transactions window will be selected
- **Clear All** – The contents of the Transactions window will be cleared

The Transaction window also allows usage of the common keyboard shortcuts Ctrl-X, Ctrl-C, Ctrl-V to cut, copy and paste from the clipboard.

CLEAR – The contents of the Transactions window will be cleared

RESET TIME – Resets the PICKit Serial clock – applicable only if Time Markers are enabled on the Configuration page.

3.7 SERIAL COMMUNICATIONS MODES

Detailed information about the entering of data and commands for specific serial communications modes are given in the following chapters.

PICKit™ Serial Analyzer User's Guide

NOTES:

Chapter 4. I²C™ Master Communications

4.1 INTRODUCTION

This chapter describes the I²C Master Communications mode. I²C data and commands can be entered using a Basic Communications window or by creating Script Commands.

It is assumed that the user is familiar with the I²C protocol. For more information see:

- The I²C-Bus Specification Version 2.1 January 2000 is available from NXP Semiconductor (formerly Philips Semiconductor) web site at http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf
- An I²C Master Communications tutorial is available on the Microchip Technology web site. Click on the links: Support -> Getting Started -> PIC MCU Tutorials -> I²C Master Mode
- Several application notes are available on the Microchip Technology web site. Click on links: Design -> App Notes -> Function: Communications -> I²C

4.2 HIGHLIGHTS

This chapter discusses:

- PICKit Serial Pin Assignments
- Selecting Communications Mode
- Configuring I²C Communications Mode
- Communications: Basic Operations
- Script Builder
- Script Execute

4.3 PICKit SERIAL PIN ASSIGNMENTS

The PICKit Serial Analyzer pin assignments for I²C Master mode are:

TABLE 4-1: PIN ASSIGNMENTS

Pin	Label	Type	Description
1	AUX1	Input/Output	Auxiliary I/O port pin No. 1
2	+V	Power	Target Power
3	GND	Power	Ground
4	SDA	Input/Output	Serial Data
5	SCL	Power	Serial Clock
6	AUX2	Input/Output	Auxiliary I/O port pin No. 2

4.4 SELECTING COMMUNICATIONS MODE

The I²C Master Communications mode is selected from the Configuration Wizard or menu bar.

Configuration Wizard – Select *PICKit Serial Analyzer > Run Configuration Wizard* from the menu bar

PICKit™ Serial Analyzer User's Guide

Menu Bar – Select *PICKit Serial Analyzer > Select Communications Mode > I²C Master*

4.5 CONFIGURING I²C COMMUNICATIONS MODE

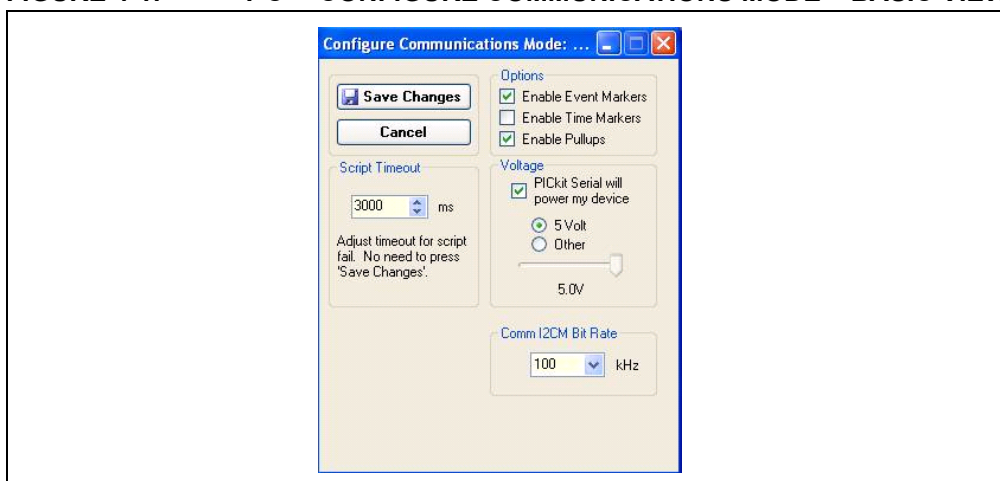
Once the communications mode has been selected, it is configured from the Configuration Wizard or menu bar.

Configuration Wizard – Select *PICKit Serial Analyzer > Run Configuration Wizard* from the menu bar

Menu Bar – Select *PICKit Serial Analyzer > Configure Communications Mode*

The Configure Mode window will open. Depending on the View selected, the Basic View (Figures 4-1) displays a minimum choice of configurations commands. In the Advanced View (Figures 4-2) displays an extended choice of configuration commands. Save the configuration by clicking on the **Save Changes** button.

FIGURE 4-1: I²C™ CONFIGURE COMMUNICATIONS MODE – BASIC VIEW



OPTIONS

- Enable Event Markers – Enable event markers
- Enable Time Markers – Enable ‘time’ stamp to accompany all event markers
- Enable Pull-ups – Enable internal 2.2 kΩ pull-ups on SDA and SCL communication lines

VOLTAGE

- PICKit Serial will power my device – Select the check box if the PICKit Serial will power the target device. The target can be powered at 5 VDC or a user selectable variable voltage.

CAUTION

Even though the voltage can be set as low as 0 VDC, it is up to the user to verify the required operating voltage of the target device.

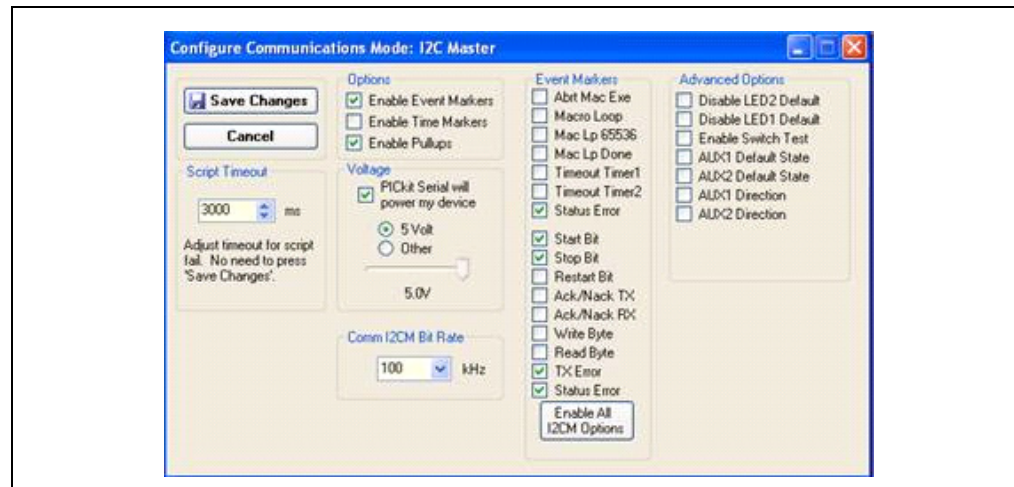
CAUTION

The USB port current limit is set to 100 mA. If the target plus PICKit Serial Analyzer exceeds this current limit, the USB port will turn off. The target may be powered externally if more power is required.

COMM I²C BIT RATE

Select the desired I²C bus bit rate using the drop down box.

FIGURE 4-2: I²C™ CONFIGURE COMMUNICATIONS MODE – ADVANCED VIEW



SCRIPT TIMEOUT

When sending scripts, the software will wait a maximum of Script Timeout ms to receive a script complete tag before issuing an error. If your I²C hardware is slow to respond or you are transferring a lot of data, you may need to increase the Script Timeout to avoid errors.

EVENT MARKERS

- Abt Mac Exe – Enable event marker: abort ‘macro’ execution
- Macro Loop – Enable event marker: top of ‘macro’ loop
- Mac Lp 65536 – Enable event marker: ‘macro’ loop count overflow (i.e., 65536)
- Mac Lp Done – Enable event marker: ‘macro’ loop iterations complete
- Timeout Timer1 – Enable event marker: Timer1 expired
- Timeout Timer2 – Enable event marker: Timer2 expired
- Status Error – Enable event marker: change in status byte

- Start Bit – Enable event marker – Start bit
- Stop Bit – Enable event marker – Stop bit
- Restart Bit – Enable event marker – Restart bit
- Ack/Nack TX – Enable event marker – Ack or Nack byte transmit
- Ack/Nack RX – Enable event marker – Ack or Nack byte received
- Write Byte – Enable event marker – write byte
- Read Byte – Enable event marker – read byte
- TX Error – Enable event marker – TX error
- Status Error – Enable event marker – change in I²C status byte

ADVANCED OPTIONS

- Disable LED2 Default – Disable default LED2 behavior (LED2 = Yellow ‘Target’ LED)
- Disable LED1 Default – Disable default LED1 behavior (LED1 = Red ‘Busy’ LED)
- Enable Switch Test – Enable low level switch test:

- Switch Off (not depressed) – blink LED1, LED2 off
- Switch ON (depressed) – blink LED2, LED1 off
- AUX1 Default State – AUX1 communication line – default state (0 | 1)
- AUX2 Default State – AUX2 communication line – default state (0 | 1)
- AUX1 Direction – AUX1 communication line – direction: 1: input, 0: output
- AUX2 Direction – AUX2 communication line – direction: 1: input, 0: output

4.6 COMMUNICATIONS: BASIC OPERATIONS

The I²C Basic Operations window can be opened by selecting:

- Communications: Basic Operations from the tool bar, or
- Communications > Basic Operations from the menu bar

The I²C Basic Operations window is shown in Figures 4-3. There are three basic communications commands, Read and Write and Receive.

Read performs a combination Write then Read commands to the target device (refer to the I²C Specification reference in **Section 4.1 “Introduction”** above). The basic structure of the command is:

- Start bit (S_)
- Slave Address[W] – Enter the slave address of the device to communicate with. The write bit should be cleared to indicate a write operation
- Word Address – Enter the word address
- Restart (RS),
- Slave Address[R] – The slave address with the write bit set will be automatically entered when the Slave Address[W] has been entered
- Byte Count – Enter the number of bytes to be read
- Stop bit (P_)

Note: The “x” indicates the value is a hexadecimal number. Clicking on “x” will toggle it to a “d” indicating that the value is a decimal number.

Write performs a write operation to the target device (refer to The I²C Specification reference in **Section 4.1 “Introduction”** above). The basic structure of the command is:

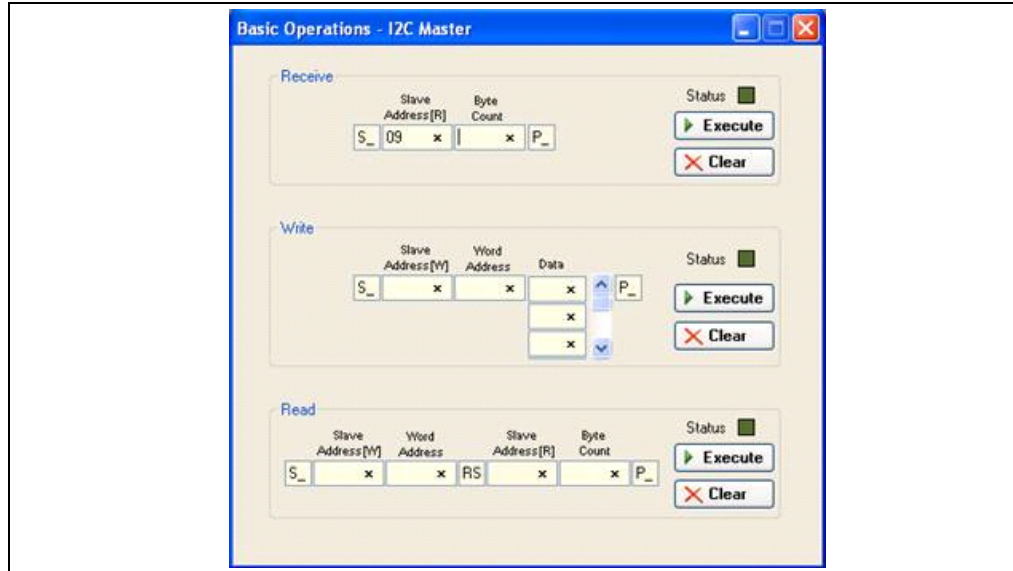
- Start bit (S_)
- Slave Address[W] – Enter the slave address of the device to communicate with. The write bit should be cleared to indicate a write operation.
- Word Address – Enter the word address
- Data – Enter up to eight bytes of data
- Stop bit (P_)

The command will be logged in the Transactions window. A listing of the command abbreviations is given in Table 4-2.

Receive issues a simple read command to the target device (refer to The I²C Specification reference in Section 4.1 “Introduction” above). The basic structure of the command is:

- Start bit (S_)
- Slave Address[W] – Enter the slave address of the device to communicate with. The write bit should be set to indicate a read operation.
- Byte Count – Enter the number of bytes to receive
- Stop bit (P_)

FIGURE 4-3: I²C™ BASIC OPERATIONS

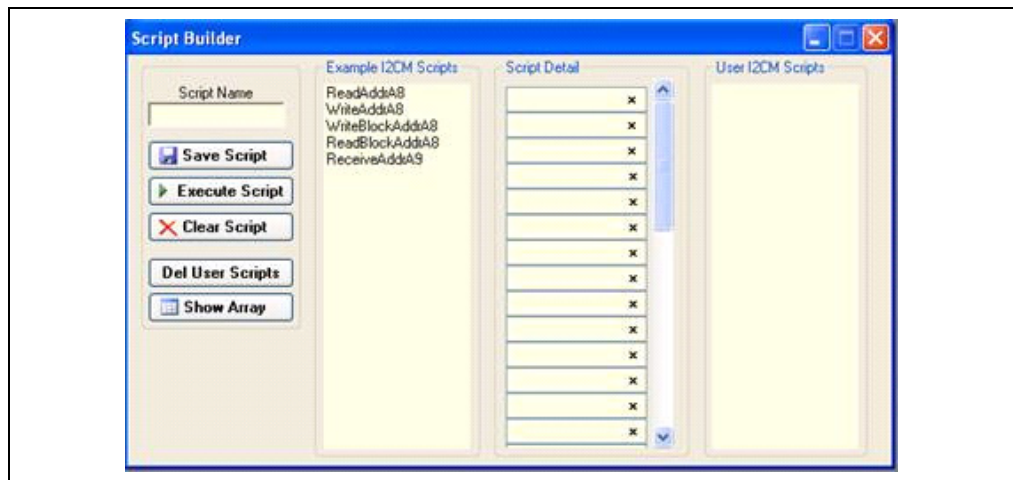


4.7 SCRIPT BUILDER

I²C commands can be combined into scripts, saved, and used over again. The Script Builder window is opened by selecting *Communications > Script > Script Builder* from the menu bar. The Script Builder is shown in Figures 4-4.

The Script Builder window is divided into four columns as shown in Figures 4-4 through 4-8.

FIGURE 4-4: I²C™ SCRIPT BUILDER



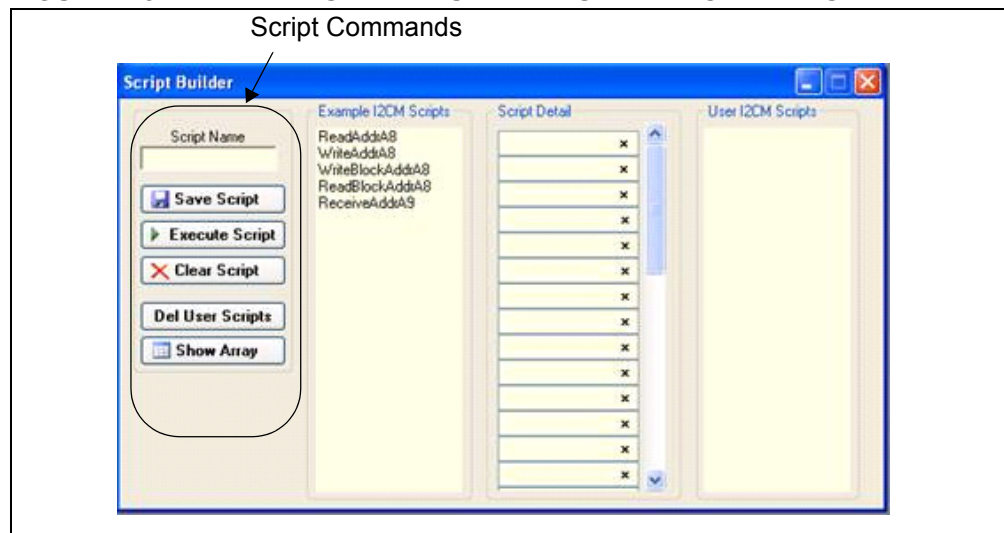
4.7.1 Script Commands

The left most column contains the Script Commands as shown in Figures 4-5.

- Script Name – Enter the name of the script
- Save Script – Saves the script
- Execute Script – Executes (performs) the script displayed in the Script Detail column
- Clear Script – Clears the Script Detail column
- Del User Scripts – Deletes scripts from the User Scripts column.
- Show Array – Displays a spreadsheet-like table in which large amounts of data

may be entered. This data can be included in the script by right clicking in a Script Detail cell and choosing "Insert Array".

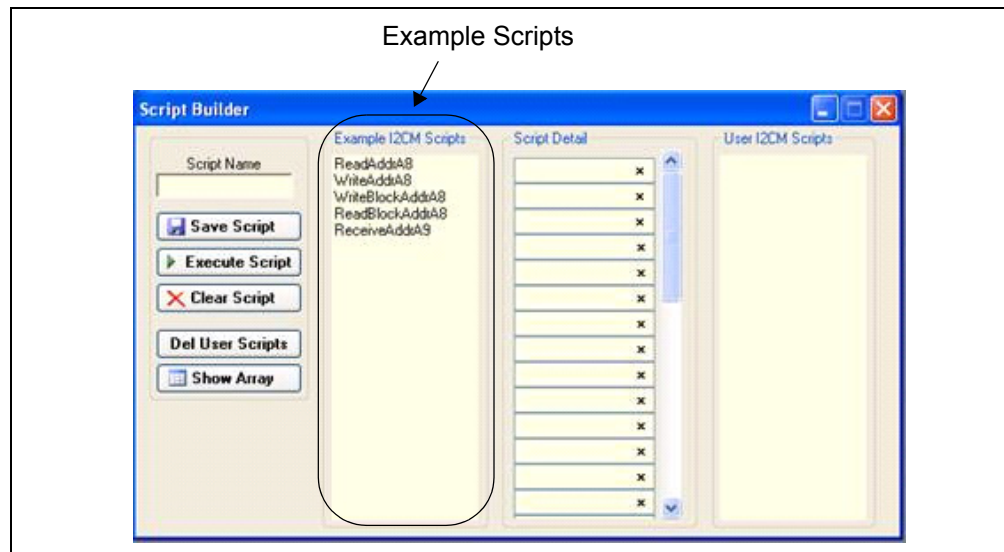
FIGURE 4-5: I²C™ SCRIPT BUILDER – SCRIPT COMMANDS



4.7.2 Example Scripts

The second column contains Example Scripts as shown in Figures 4-6. These can be studied to learn how to create or to edit custom scripts. To load the example script into the Script Detail column, either double click or right click and select from the local menu.

FIGURE 4-6: I²C™ SCRIPT BUILDER – EXAMPLE SCRIPTS



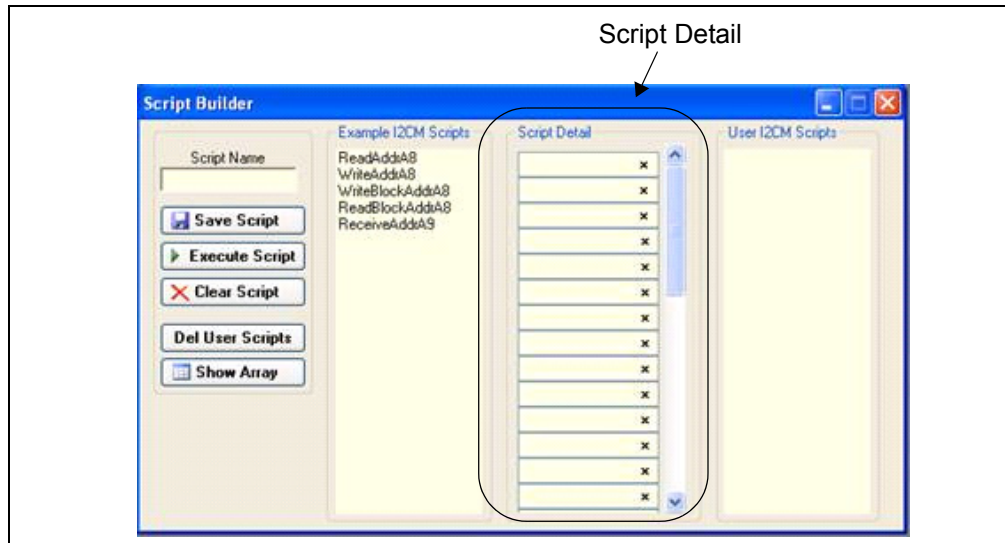
4.7.3 Script Detail

The third column contains Script Detail as shown in Figures 4-7. This column is used to create the script or view an existing script. More information about creating a custom script is discussed in **Section 4.7.5 "Creating A Script"**.

To load a user script from the User Scripts column into the Script Detail column, the user can double click or right click and select from the local menu.

Note: The “x” indicates the value is a hexadecimal number. Clicking on “x” will toggle it to a “d” indicating that the value is a decimal number.

FIGURE 4-7: I²C™ SCRIPT BUILDER – SCRIPT DETAIL



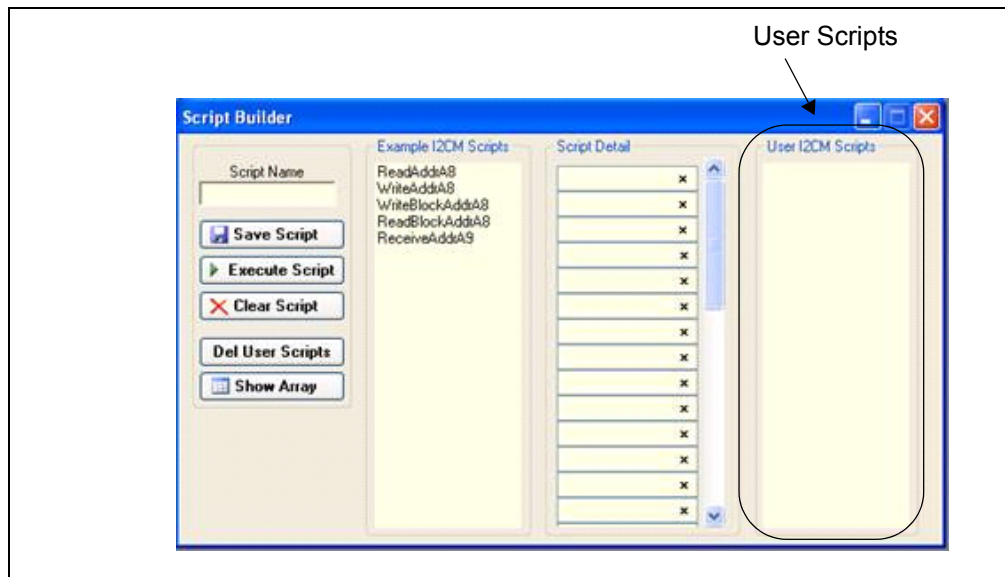
4.7.4 User Scripts

The fourth column contains User Scripts as shown in Figures 4-8. User scripts that are created, named, and saved are displayed in the User Scripts column.

To load a user script from the User Scripts column into the Script Detail column, the user can double click or right click and select from the local menu.

User Scripts can be deleted by right clicking and selecting Delete Script from the local menu.

FIGURE 4-8: I²C™ SCRIPT BUILDER – USER SCRIPTS



4.7.5 Creating A Script

Scripts are created by placing the cursor into the Script Detail column and right clicking. A local menu will be displayed as shown in Figures 4-9. Select from the choice of commands or script macro commands.

The sequence of macro commands are executed from top to bottom. Macro commands are entered by right clicking in the box and selecting from the local menu as shown in Figures 4-9.

Macro commands are entered according to the sequence of events as defined by the I²C bus protocol. Studying the example scripts is a good way to learn the sequence of events. The example scripts can also be modified and saved under a different name.

CAUTION

The choice of macro commands is very flexible. Therefore, the correctness of the script has to be verified by the user. The PICKit Serial Analyzer program does not verify the correctness of the script.

A complete listing of the available macro commands is given in Table 4-2. The macro command abbreviation will be displayed in the Transactions Window. The Transactions window keeps a running log of the commands and data sent to and from the target device.

FIGURE 4-9: I²C™ SCRIPT BUILDER – CREATING A SCRIPT

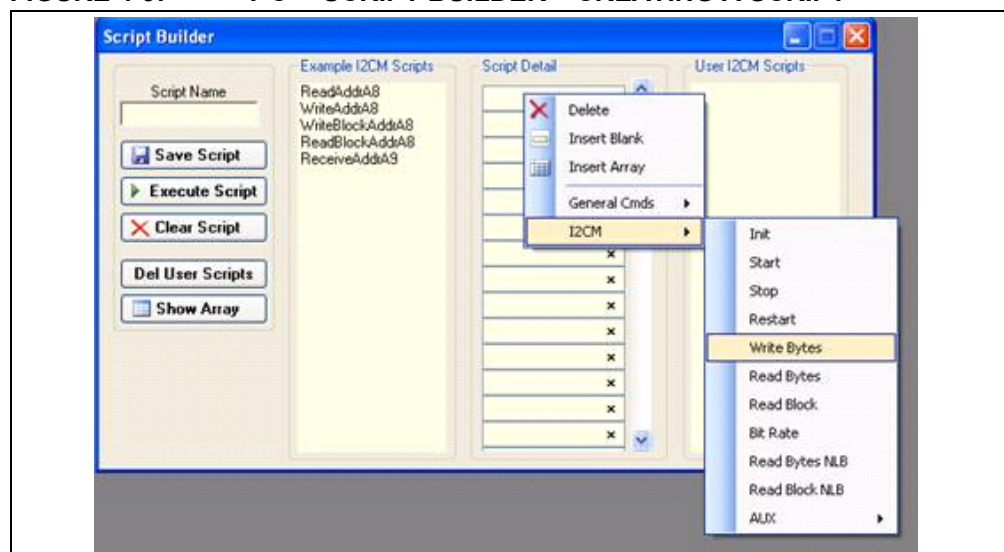


TABLE 4-2: I²C™ SCRIPT MACRO COMMAND

Macro Command	Command Abbreviation	Description
I2CINIT	[I_]	I ² C™ Initialization
I2CSTART	[S_]	I ² C™ Start
I2CSTOP	[P_]	I ² C™ Stop
I2CRESTART	[RS]	I ² C™ Restart
I2CWRTBYT	[W_]	I ² C™ Write Bytes. Next byte is the byte count, followed by the data.
I2CRDBYT	[R_]	I ² C™ Read Bytes. Next byte is the byte count.

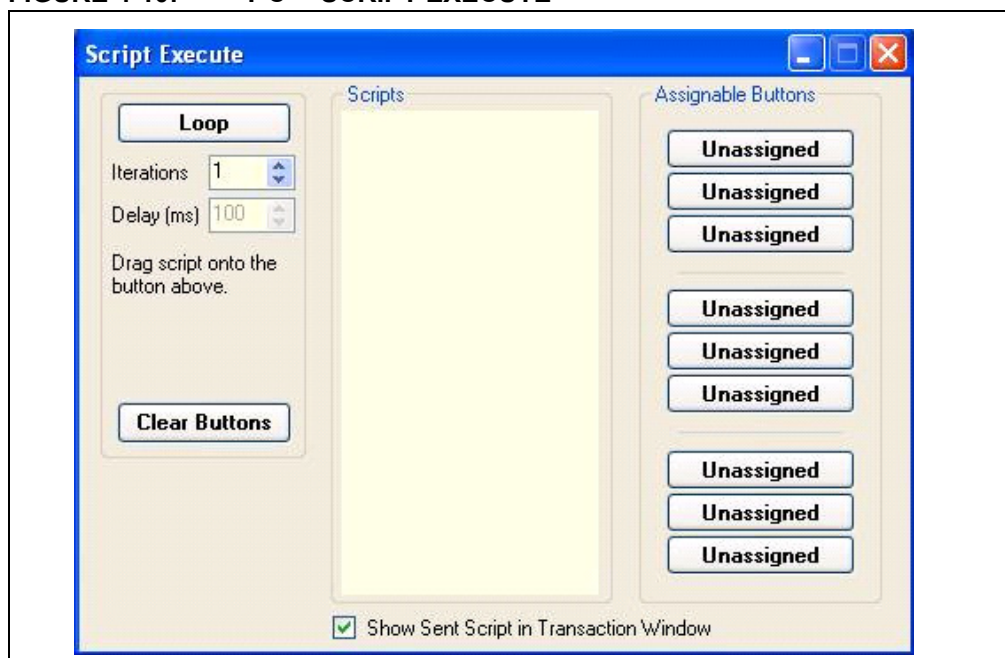
TABLE 4-2: I²C™ SCRIPT MACRO COMMAND (CONTINUED)

I2CRDBLK	[RB]	I ² C™ Read Block
I2CBITRATE	[BR]	Set I ² C™ Bit Rate - min:0 = 35k, max:127 = 100k. Next byte is the bit rate.
I2CRESET	[RE]	Reset MSSP module
I2CRDBYTNLB	[RN]	Read bytes - NACK last byte. Next byte is the byte count
I2CRDBLKNLB	[RBN]	Read block - NACK last byte
I2CAUX1RST	[A1RST]	Reset AUX1
I2CAUX1SET	[A1RST]	Set AUX1
I2CAUX1OUT	[A1OUT]	Set AUX1 direction to Output
I2CAUX1IN	[A1IN]	Set AUX1 direction to Input
I2CAUX1W0	[A1W0]	AUX1 Wait 0
I2CAUX1W1	[A1W1]	AUX1 Wait 1
I2CAUX2RST	[A2RST]	Reset AUX2
I2CAUX2SET	[A2RST]	Set AUX2
I2CAUX2OUT	[A2OUT]	Set AUX2 direction to Output
I2CAUX2IN	[A2IN]	Set AUX2 direction to Input
I2CAUX2W0	[A2W0]	AUX2 Wait 0
I2CAUX2W1	[A2W1]	AUX2 Wait 1

4.8 SCRIPT EXECUTE

The Script Execute window is shown in Figures 4-10. Once scripts are created using the Script Builder, they can be assigned to buttons in the Script Execute window. This makes a convenient window to execute multiple scripts either individually or iteratively. Script executing will be logged in the Transactions window. The Script Execute window is opened by selecting *Communications > Script > Script Execute* from the menu bar.

FIGURE 4-10: I²C™ SCRIPT EXECUTE



4.8.1 Assignable Buttons

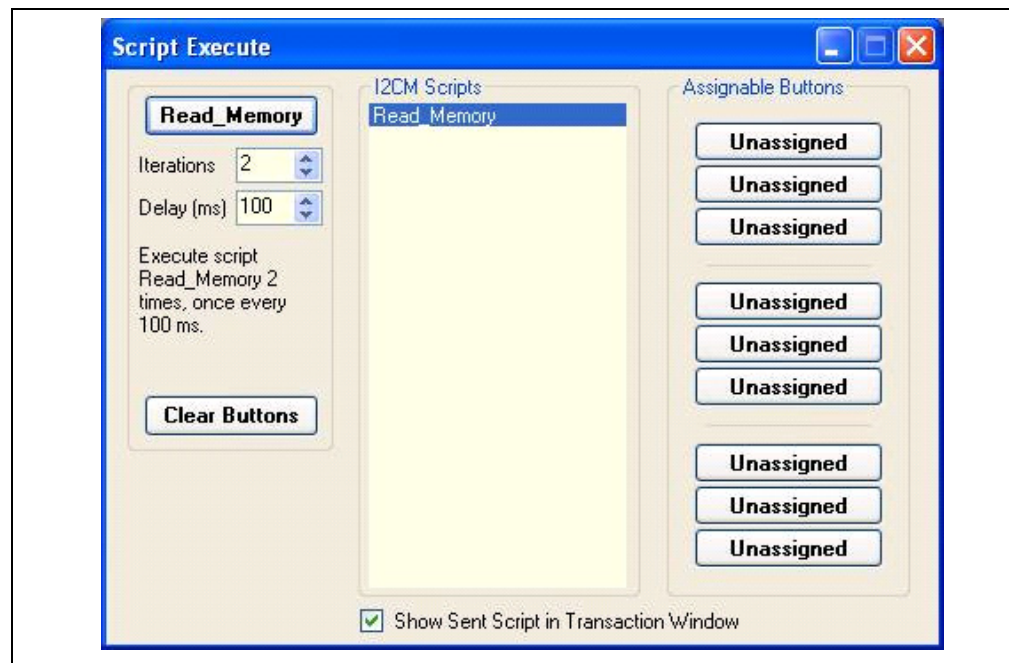
User created scripts will be displayed in the central I²C Scripts column. To assign a script to a button, click on the script name and drag it to the desired Assignable Buttons in the right column. The script will be executed once each time the button is clicked.

The Assignable Buttons can be cleared by clicking on the **Clear Buttons** button.

4.8.2 Iteration

Scripts can be executed a user defined number of times at a specified interval of time. Figure 4-11 shows an example. A script named Read_Memory has been assigned to the **Iteration** button in the left column. The number of iterations are entered in the Iterations box and the delay in millisecond in the Delay box. A summary of the iterations is displayed in the left column. The macro is executed when the **Iteration** button is clicked.

FIGURE 4-11: I²C™ SCRIPT EXECUTE – EXAMPLE



Chapter 5. I²C™ Slave Communications

5.1 INTRODUCTION

This chapter describes I²C Slave Communications. The I²C Slave mode allows you to mimic an I²C slave by responding to standard Read, Write, and Receive commands.

You can mimic an I²C slave device from either the PICKit Serial Analyzer PC interface (Interactive mode), or from the PICKit Serial Analyzer itself (Auto mode).

Additionally, you can run the Interactive mode from either the Basic Communications page, or from the I²C Slave Profile page.

It is assumed that the user is familiar with the I²C protocol. For more information see:

- The I²C-Bus Specification Version 2.1 January 2000 is available from NXP Semiconductor (formerly Philips Semiconductor) web site at http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf
- An I²C Master Communications tutorial is available on the Microchip Technology web site. Click on the links: Support -> Getting Started -> PIC MCU Tutorials -> I²C Master Mode
- Several application notes are available on the Microchip Technology web site. Click on links: Design -> App Notes -> Function: Communications -> I²C

5.2 HIGHLIGHTS

This chapter discusses:

- PICKit Serial Pin Assignments
- Selecting Communications Mode
- Configuring I²C Communications Mode
- Communications: Basic Operations
- I²C Profile Generator

5.3 PICKit SERIAL PIN ASSIGNMENTS

The PICKit Serial Analyzer pin assignments for I²C Slave mode are listed in the following table:

TABLE 5-1: PIN ASSIGNMENTS

Pin	Label	Type	Description
1	Unused	–	–
2	+V	Power	Target Power
3	GND	Power	Ground
4	SDA	Input/Output	Serial Data
5	SCL	Power	Serial Clock
6	Unused	–	–

PICkit™ Serial Analyzer User's Guide

5.4 SELECTING COMMUNICATIONS MODE

The I²C Master Communications mode is selected from the Configuration Wizard or menu bar.

Configuration Wizard – Select *PICkit Serial Analyzer > Run Configuration Wizard* from the menu bar

Menu Bar – Select *PICkit Serial Analyzer > Select Communications Mode > I²C Slave*

5.5 CONFIGURING I²C SLAVE COMMUNICATIONS MODE

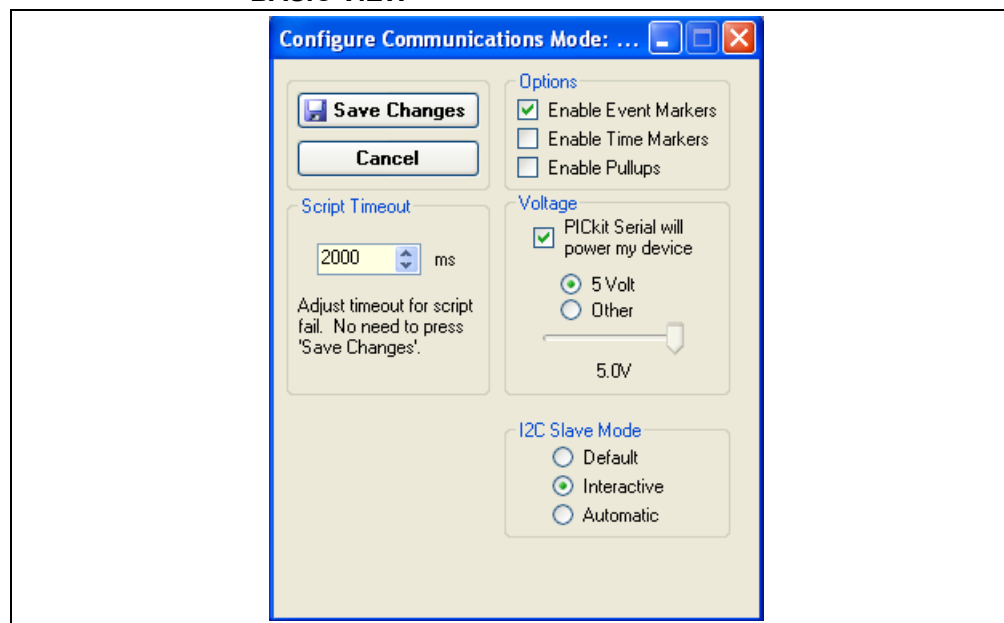
Once the communications mode has been selected, it is configured from the Configuration Wizard or menu bar.

Configuration Wizard – Select *PICkit Serial Analyzer > Run Configuration Wizard* from the menu bar

Menu Bar – Select *PICkit Serial Analyzer > Configure Communications Mode*

The Configure Mode window will open. Depending on the View selected, the Basic View (Figure 5-1) displays a minimum choice of configurations commands. The Advanced View (Figure 5-2) displays an extended choice of configuration commands. Save the configuration by clicking on the **Save Changes** button.

FIGURE 5-1: I²C™ SLAVE CONFIGURE COMMUNICATIONS MODE – BASIC VIEW



OPTIONS

- Enable Event Markers – Enable event markers
- Enable Time Markers – Enable 'time' stamp to accompany all event markers
- Enable Pull-ups – Enable internal 2.2 k Ω pull-ups on SDA and SCL communication lines

VOLTAGE

- PICkit Serial will power my device – Select the check box if the PICkit Serial will power the target device. The target can be powered at 5 VDC or a user selectable variable voltage.

I²C SLAVE MODE

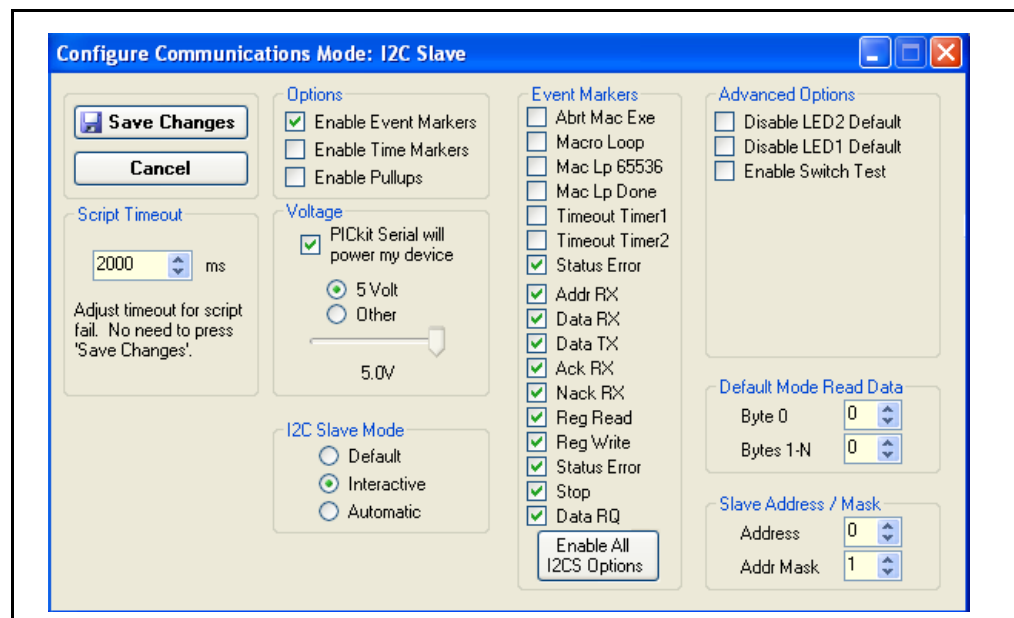
There are three modes of I²C slave operation:

- **DEFAULT:** Basic/mechanical mode of operation in which the PKSA blindly accepts any/all 'write' data and provides canned/default 'read' data in response to any enabled I²C address and all device addresses.
- **INTERACTIVE:** This mode allows the host to orchestrate I²C transactions in 'real time'. This necessarily requires the host to provide 'read' and 'receive' data as needed while the PKSA holds the I²C bus clock line (waiting). 'Write' data is reported to the host via transaction event tags.
- **AUTO:** In AUTO mode the PKSA operates autonomously as defined by a dynamic "SLAVE PROFILE" table stored in PKSA RAM. At any time the host can read and/or update the table as needed.

SCRIPT TIMEOUT

When the slave responds to Read and Receive queries the software will wait a maximum of Script Timeout ms to receive a script complete tag before issuing an error. If your slave profile is responding with large amounts of data, you may need to increase the Script Timeout to avoid errors.

FIGURE 5-2: I²C™ SLAVE CONFIGURE COMMUNICATIONS MODE – ADVANCED VIEW



EVENT MARKERS

- Abt Mac Exe – Enable event marker: abort 'macro' execution
- Macro Loop – Enable event marker: top of 'macro' loop
- Mac Lp 65536 – Enable event marker: 'macro' loop count overflow (i.e., 65536)
- Mac Lp Done – Enable event marker: 'macro' loop iterations complete
- Timeout Timer1 – Enable event marker: Timer1 expired
- Timeout Timer2 – Enable event marker: Timer2 expired
- Status Error – Enable event marker: change in status byte
- Addr RX – Enable event marker – Slave address received
- Data RX – Enable event marker – Data received
- Data TX – Enable event marker – Data transmitted
- Ack RX – Enable event marker – Ack byte received

PICkit™ Serial Analyzer User's Guide

- Nack RX – Enable event marker – Nack byte received
- Reg Read – Enable event marker – Register Read
- Reg Write – Enable event marker – Register Write
- Status Error – Enable event marker – change in I²C status byte
- Stop – Enable event marker – I²C Stop
- Data RQ – Enable event marker – Data request

Note: Disabling any of the event markers may cause the software to not respond correctly, or at all.

ADVANCED OPTIONS

- Disable LED2 Default – Disable default LED2 behavior (LED2 = Yellow 'Target' LED)
- Disable LED1 Default – Disable default LED1 behavior (LED1 = Red 'Busy' LED)
- Enable Switch Test – Enable low level switch test:

DEFAULT MODE READ DATA

- Sets byte 0 and following byte response for Default mode

SLAVE ADDRESS / MASK

- Sets the I²C slave address to which the PICkit Serial Analyzer will respond

Note: The Address Mask is not currently implemented due to hardware limitations. It is left in the software for possible future use.

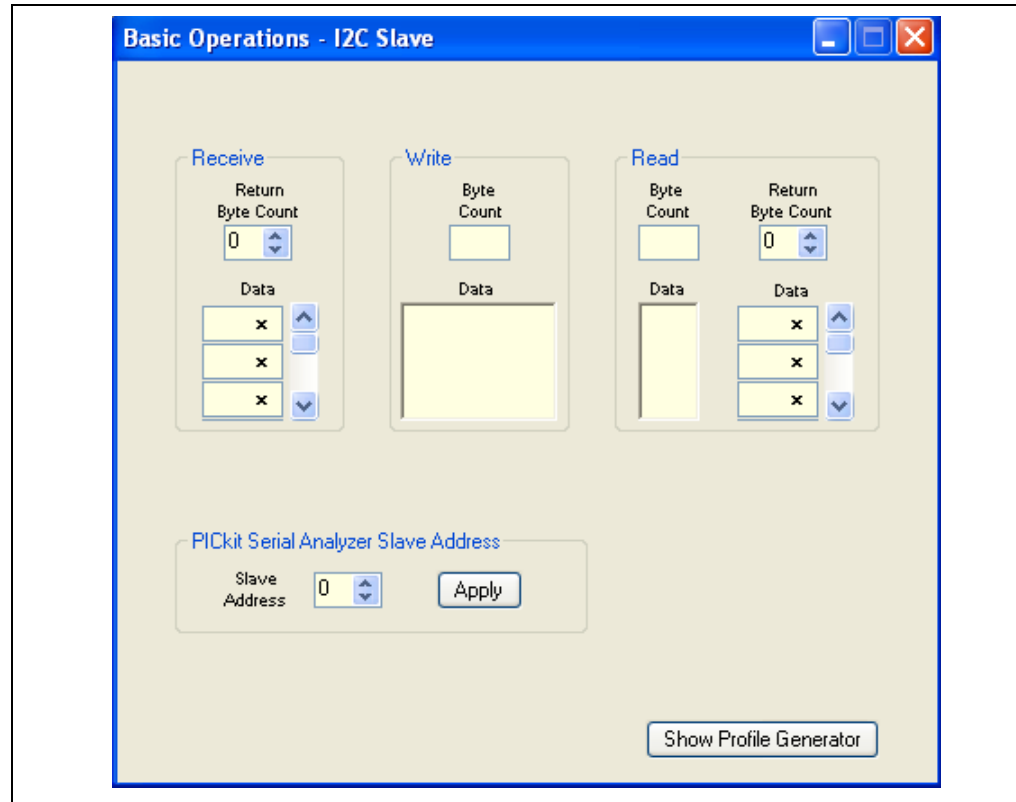
5.6 COMMUNICATIONS: BASIC OPERATIONS

The I²C Slave Basic Operations window can be opened by selecting:

- *Basic Operations* from the tool bar, or
- *Communications > Basic Operations* from the menu bar

The I²C Slave Basic Operations window is shown in Figures 5-3. There are three basic communications commands, Receive, Write and Read.

FIGURE 5-3: I²C™ SLAVE – BASIC OPERATIONS



Receive responds to a basic Receive request from the master. In Return Byte Count, enter the number of bytes a Receive request will respond with, then enter the data to be returned in the Data array. The Basic page limits you to eight bytes of returned data.

Note: The "x" indicates the value is a hexadecimal number. Clicking on "x" will toggle it to a "d" indicating that the value is a decimal number.

Write simply displays the byte count and data written by the master.

Read responds to a basic Read request by the master. In Return Byte Count, enter the number of bytes a Read request will respond with, then enter the data to be returned in the Data array. The Basic page limits you to eight bytes of returned data.

Byte Count will display the number of bytes between the slave read address and the slave write address. Data will display the data between the slave read address and the slave write address.

PICkit Serial Slave Address set the slave address here you wish the software to respond to, then press the Apply button. A confirmation message should appear on the Transactions window.

PICKit™ Serial Analyzer User's Guide

Show Profile Generator. Press this to close the Basic operations page and open the I²C Slave Profile Generator page.

Note: The Basic operations page and the Profile Generator page cannot both be opened at the same time due to dll conflict issues.

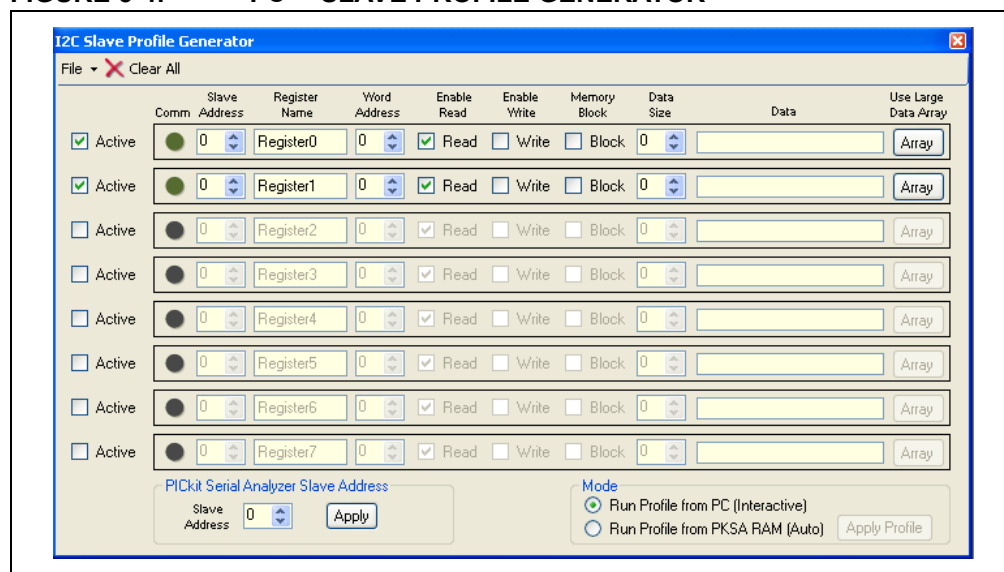
5.7 COMMUNICATIONS: PROFILE GENERATOR

The I²C Slave Profile Generator allows you to mimic an I²C slave device from either the PICKit Serial Analyzer PC interface, or from the PICKit Serial Analyzer itself. The trade off between the two methods is flexibility vs. response time. Mimicking a device from the PC allows for a great deal of run time flexibility, while running from the PICKit Serial Analyzer decreases response time. You can mimic up to eight registers in a profile.

DISPLAYING THE I²C SLAVE PROFILE GENERATOR

The profile generator may be opened by either pressing the 'Show Profile Generator' button on the I²C Basic Operations page, or by the menu item *Communications > I²C Slave Profile Generator*. Figure 5-4 shows the Slave Profile Generator with the first two registers activated.

FIGURE 5-4: I²C™ SLAVE PROFILE GENERATOR



CREATING A PROFILE

Create a profile by filling in *Inputs* for each register:

- **Active** – Allows you to activate or deactivate a register. This must be active before you can fill in the rest of the data. The purpose of this checkbox is to allow run time activation or deactivation of a register.
- **Comm** – Flashes when the register is activated (read from or written to).
- **Slave Address** – Hex representation of the register's slave address.
- **Register Name** – Enter a register name. This is for reference only and does not affect operation.
- **Word Address** – Hex representation of the word address of the register (will be forced to zero if Memory Block is selected).
- **Enable Read** – Determines whether or not the register is read accessible.
- **Enable Write** – Determines whether or not the register is write accessible.

- Memory Block – Check if the register is to be treated as a contiguous block of memory, such as an EE array.
- Data Size – Number of bytes in the register. If the register is not a Memory Block, this many bytes will be returned when the register is read. If the register is a Memory Block, the profile generator will return one byte at a time until it is Nacked.
- Data – Stores the data belonging to the register (Hex format)
- Use Large Data Array – If the data in the register is larger than 8 bytes, press this button to display a 256 byte array into which you may enter data.

Once the parameters are entered, they can be saved as an I²C Slave Profile file (*.pfl).

- File->Open – Open an existing profile (*.pfl)
- File->Save – Save or replace a profile (*.pfl)
- Clear All – Resets (clears) all registers

PICKit SERIAL SLAVE ADDRESS

In the groupbox 'PICKit Serial Analyzer Slave Address' enter the hex representation of the slave address you want the PICKit Serial Analyzer to respond to, then press the 'Apply' button. This writes the address into the Analyzer firmware.

MODE

If you wish to run the slave profile from the PC (GUI) there is nothing more to do. If you wish to run it from the PICKit Serial Analyzer, press the appropriate radio button in the 'Mode' groupbox.

Note: If you are running the profile from the PICKit Serial Analyzer and you wish to change the profile, you must press the 'Apply Profile' button to load the changes into the firmware.

Note: When running the profile from the PICKit Serial Analyzer instead of the PC, you are limited to 255 data and configuration bytes (profile length). If you attempt to write too large of a profile to the PICKit Serial Analyzer, an error message will be sent to the Transactions window telling you your profile length.

Chapter 6. Lin Communications

6.1 INTRODUCTION

This chapter describes the LIN Communications mode. LIN data and commands can be entered using a Basic Communications window.

To use LIN communications, you must attach the LIN Adapter to the PICKit Serial Analyzer and supply it power prior to attempting any communications.

It is assumed that the user is familiar with the LIN protocol. For more information see: *LIN bus Specification Ver. 1.2*: <http://www.lin-subbus.de/>

6.2 HIGHLIGHTS

This chapter discusses:

- PICKit Serial Pin Assignments
- Selecting Communications Mode
- Configuring LIN Communications Mode
- Communications: Basic Operations

6.3 PICKit SERIAL PIN ASSIGNMENTS

The PICKit Serial Analyzer pin assignments for LIN mode are:

TABLE 6-1: PIN ASSIGNMENTS

Pin	Label	Type	Description
1	TX	Output	Transmit
2	Unused	–	–
3	GND	Power	Ground
4	CS/Wake	Input	Chip Select / Wake
5	Fault/TXE	Input/Output	Fault / TXE
6	RX	Input	Receive

6.4 SELECTING COMMUNICATIONS MODE

The LIN Communications mode is selected from the Configuration Wizard or menu bar.

Configuration Wizard – Select *PICKit Serial Analyzer > Run Configuration Wizard* from the menu bar

Menu Bar – Select *PICKit Serial Analyzer > Select Communications Mode > LIN*

6.5 CONFIGURING LIN SLAVE COMMUNICATIONS MODE

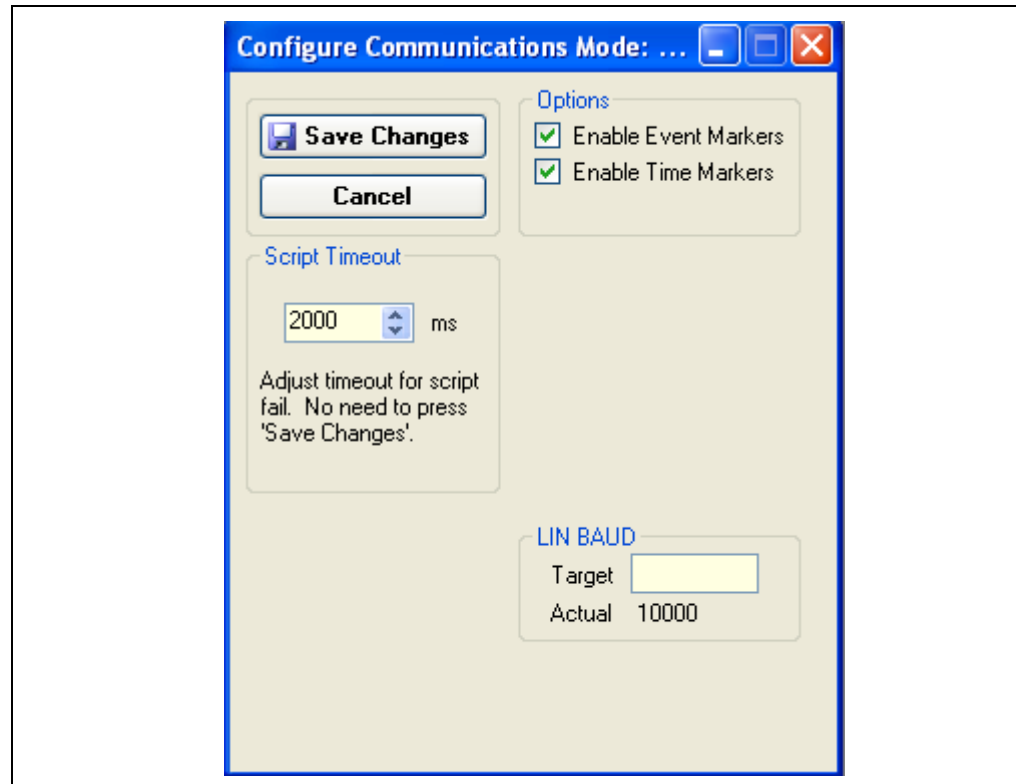
Once the communications mode has been selected, it is configured from the Configuration Wizard or menu bar.

Configuration Wizard – Select *PICkit Serial Analyzer > Run Configuration Wizard* from the menu bar

Menu Bar – Select *PICkit Serial Analyzer > Configure Communications Mode*

The Configure Mode window will open. Depending on the View selected, the Basic View (Figure 6-1) displays a minimum choice of configurations commands. In the Advanced View (Figure 6-2) displays an extended choice of configuration commands. Save the configuration by clicking on the **Save Changes** button.

FIGURE 6-1: LIN CONFIGURE COMMUNICATIONS MODE – BASIC VIEW



OPTIONS

- Enable Event Markers – Enable event markers
- Enable Time Markers – Enable ‘time’ stamp to accompany all event markers

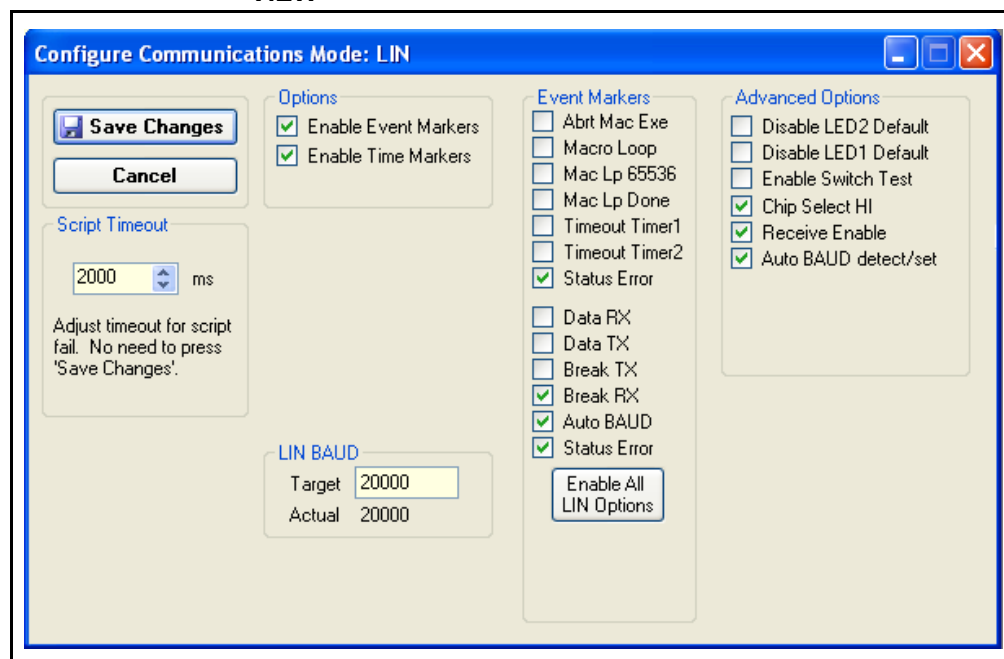
LIN BAUD

Enter desired baud rate here. You are limited to a range of 1000 to 20000.

SCRIPT TIMEOUT

When transmitting frames, the software will wait a maximum of Script Timeout ms to receive a script complete tag before issuing an error. If your LIN hardware is slow to respond, you may need to increase the Script Timeout to avoid errors.

FIGURE 6-2: LIN CONFIGURE COMMUNICATIONS MODE – ADVANCED VIEW



EVENT MARKERS

- Abt Mac Exe – Enable event marker: abort 'macro' execution
- Macro Loop – Enable event marker: top of 'macro' loop
- Mac Lp 65536 – Enable event marker: 'macro' loop count overflow (i.e., 65536)
- Mac Lp Done – Enable event marker: 'macro' loop iterations complete
- Timeout Timer1 – Enable event marker: Timer1 expired
- Timeout Timer2 – Enable event marker: Timer2 expired
- Status Error – Enable event marker: change in status byte
- Data RX – Enable event marker – Data received
- Data TX – Enable event marker – Data transmitted
- Break TX – Enable event marker – Break transmitted
- Break RX – Enable event marker – Break Received
- Auto Baud – Enable event marker – Automatic Baud Rate Selection
- Status Error – Enable event marker – change in LIN status byte

Note: Disabling any of the event markers may cause the software to not respond correctly, or at all.

ADVANCED OPTIONS

- Disable LED2 Default – Disable default LED2 behavior (LED2 = Yellow 'Target' LED)
- Disable LED1 Default – Disable default LED1 behavior (LED1 = Red 'Busy' LED)
- Enable Switch Test – Enable low level switch test:
- Chip Select Hi – Sets LIN Adapter Chip Select Hi
- Receive Enable – Enables f/w to receive data
- Auto Baud Detect/Set – Enables Auto Baud detections and setting

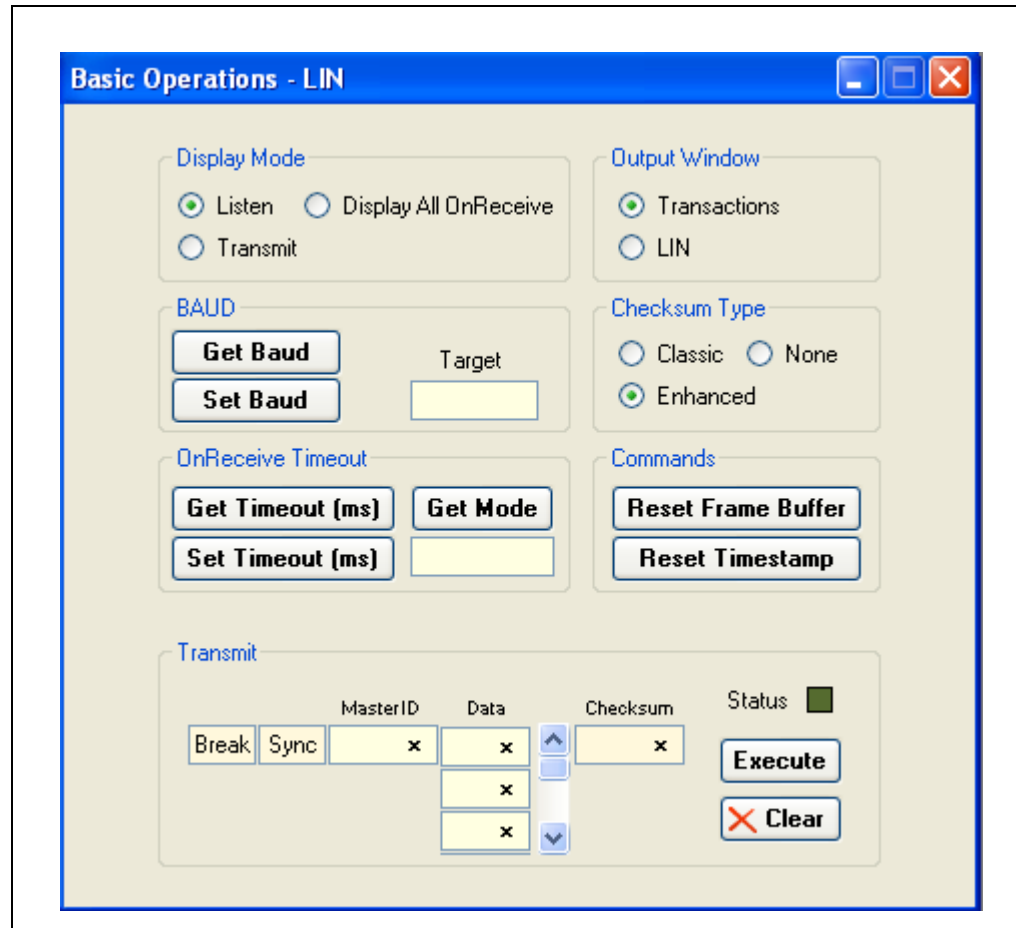
6.6 COMMUNICATIONS: BASIC OPERATIONS

The LIN Basic Operations window can be opened by selecting:

- *Basic Operations* from the tool bar, or
- *Communications > Basic Operations* from the menu bar

The LIN Basic Operations window is shown in Figure 6-3.

FIGURE 6-3: LIN BASIC OPERATIONS



Display Mode: Determines when OnReceive data is displayed. Listen will display when the completed frame is different than the last frame, Transmit will not display data, and Display All will display all OnReceive data.

Output Window: Toggles between displaying data on the Transactions window and a specialized LIN display.

BAUD: Allows the user to get and set the baud rate.

Checksum Type: Toggles between Classic, Enhanced or no checksum.

Commands: Reset Frame Buffer forces a manual reset of the working frame in the DLL. Reset Timestamp forces the next frame to be received to start at time 0.0 seconds.

PICKit™ Serial Analyzer User's Guide

Transmit. Allows the user to transmit a frame. Enter the MasterID, and frame data if any. The checksum is automatically calculated. Press the Execute button to transmit the frame. The Clear button will clear the contents of the MasterID and the data textboxes.

Note: The “x” indicates the value is a hexadecimal number. Clicking on “x” will toggle it to a “d” indicating that the value is a decimal number.

Chapter 7. SPI and Microwire Master Communications

7.1 INTRODUCTION

This chapter describes the SPI and Microwire Master Communications mode. Microwire is a subset of the SPI protocol. Data and commands can be entered using a Basic Communications window or by creating Script Commands.

It is assumed that the user is familiar with the SPI protocol. For more information see: An SPI tutorial is available on the Microchip Technology web site. Click on the links: Support -> Getting Started -> PIC MCU Tutorials -> SPI - PICmicro Serial Peripheral Interface

Several application notes are available on the Microchip Technology web site. Click on links: Design -> App Notes -> Function: Communications -> SPI

7.2 HIGHLIGHTS

This chapter discusses:

- PICKit Serial Analyzer Pin Assignments
- Selecting Communications Mode
- Configuring SPI Communications Mode
- Communications: Basic Operations
- Script Builder
- Script Execute

7.3 PICKit™ SERIAL ANALYZER PIN ASSIGNMENTS

The PICKit Serial Analyzer pin assignments for SPI Master mode are:

TABLE 7-1: PIN ASSIGNMENTS

Pin	Label	Type	Description
1	\overline{CS}	Output	Chip Select (Active Low)
2	+V	Power	Target Power
3	GND	Power	Ground
4	SDI	Input	Serial Data In (with respect to the PICKit Serial Analyzer)
5	SCK	Output	Serial Clock
6	SDO	Output	Serial Data Out (with respect to the PICKit Serial Analyzer)

7.4 SELECTING COMMUNICATIONS MODE

The SPI Master Communications mode is selected from the Configuration Wizard or menu bar.

Configuration Wizard – Select *PICKit Serial Analyzer > Run Configuration Wizard* from the menu bar

PICKit™ Serial Analyzer User's Guide

Menu Bar – Select *PICKit Serial Analyzer > Select Communications Mode > SPI Master*

7.5 CONFIGURATING SPI COMMUNICATIONS MODE

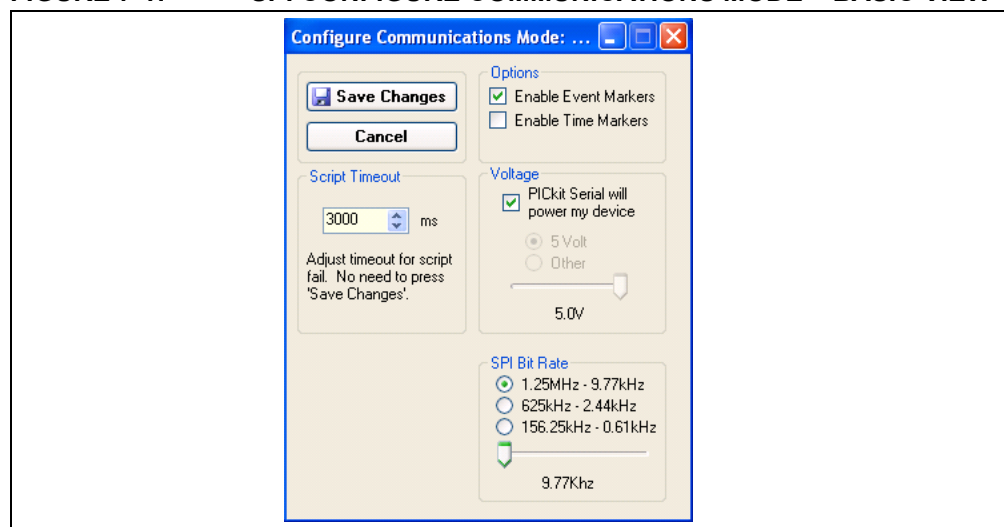
Once the communications mode has been selected, it is configured from the Configuration Wizard or menu bar.

Configuration Wizard – Select *PICKit Serial Analyzer > Run Configuration Wizard* from the menu bar

Menu Bar – Select *PICKit Serial Analyzer > Configure Communications Mode*

The Configure Mode window will open. Depending on the View selected, the Basic View (Figure 7-1) displays a minimum choice of configurations commands. In the Advanced View (Figure 7-2) displays an extended choice of configuration commands. Save the configuration by clicking on the **Save Changes** button.

FIGURE 7-1: SPI CONFIGURE COMMUNICATIONS MODE – BASIC VIEW



OPTIONS

- Enable Event Markers – Enable event markers
- Enable Time Markers – Enable ‘time’ stamp to accompany all event markers

VOLTAGE

- PICKit Serial will power my device – Select the check box if the PICKit Serial will power the target device. The target can be powered at 5 VDC or a user selectable variable voltage.

CAUTION

Even though the voltage can be set as low as 0 VDC, it is up to the user to verify the required operating voltage of the target device.

CAUTION

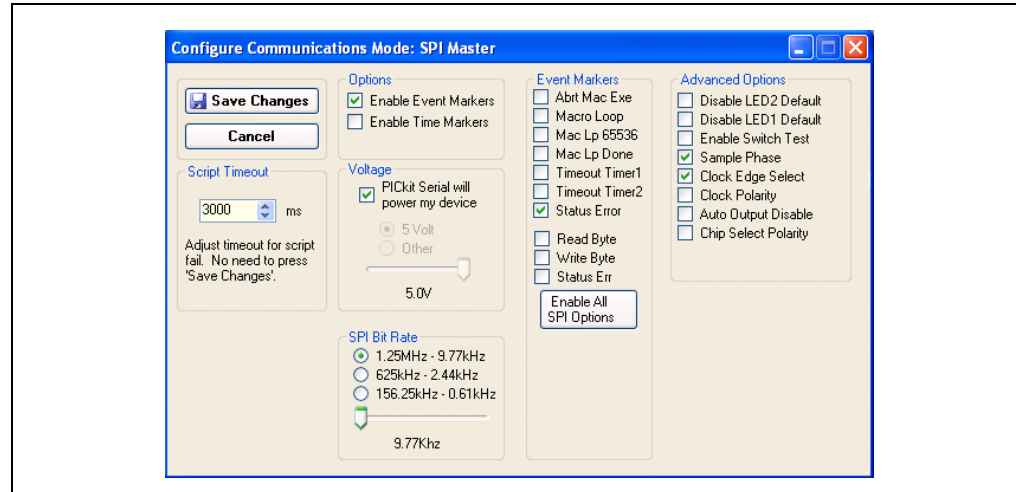
The USB port current limit is set to 100 mA. If the target plus PICKit Serial Analyzer exceeds this current limit, the USB port will turn off. The target may be powered externally if more power is required.

SPI and Microwire Master Communications

SPI BIT RATE

Select the desired SPI bit rate by selecting the radio button for the desired range and then selecting the bit rate using the slider.

FIGURE 7-2: SPI CONFIGURE COMMUNICATIONS MODE – ADVANCED VIEW



EVENT MARKERS

- Abt Mac Exe – Enable event marker: abort ‘macro’ execution
- Macro Loop – Enable event marker: top of ‘macro’ loop
- Mac Lp 65536 – Enable event marker: ‘macro’ loop count overflow (i.e., 65536)
- Mac Lp Done – Enable event marker: ‘macro’ loop iterations complete
- Timeout Timer1 – Enable event marker: Timer1 expired
- Timeout Timer2 – Enable event marker: Timer2 expired
- Status Error – Enable event marker: change in status byte
- Write Byte – Enable event marker – write byte
- Read Byte – Enable event marker – read byte
- Status Err – Enable event marker – change in SPI status byte

ADVANCED OPTIONS

- Disable LED2 Default – Disable default LED2 behavior (LED2 = Yellow ‘Target’ LED)
- Disable LED1 Default – Disable default LED1 behavior (LED1 = Red ‘Busy’ LED)
- Enable Switch Test – Enable low level switch test:
 - Switch Off (not depressed) – blink LED1, LED2 off
 - Switch ON (depressed) – blink LED2, LED1 off
- Sample Phase – SPI transaction configuration: Sample phase
- Clock Edge Select – SPI transaction configuration: Clock Edge
- Clock Polarity – SPI transaction configuration: Clock Polarity
- Auto Output Disable – Disables output during input. Allows the SDI lines and the SDO lines to be shorted for 3-wire communication.

7.6 COMMUNICATIONS: BASIC OPERATIONS

The SPI Basic Operations window can be opened by selecting:

- *Communications: Basic Operations* from the tool bar, or
- *Communications > Basic Operations* from the menu bar

The SPI Basic Operations window is shown in Figure 7-3. The Basic Operations window is organized into five columns. Individual columns are enabled by clicking on the Enable check box.

The **Send** button indicates that the column boxes are used to enter data bytes that will be transmitted to the target device. Clicking on the **Send** button toggles the column mode to Rcv (Receive) and the number of received bytes is entered as shown in Figure 7-4.

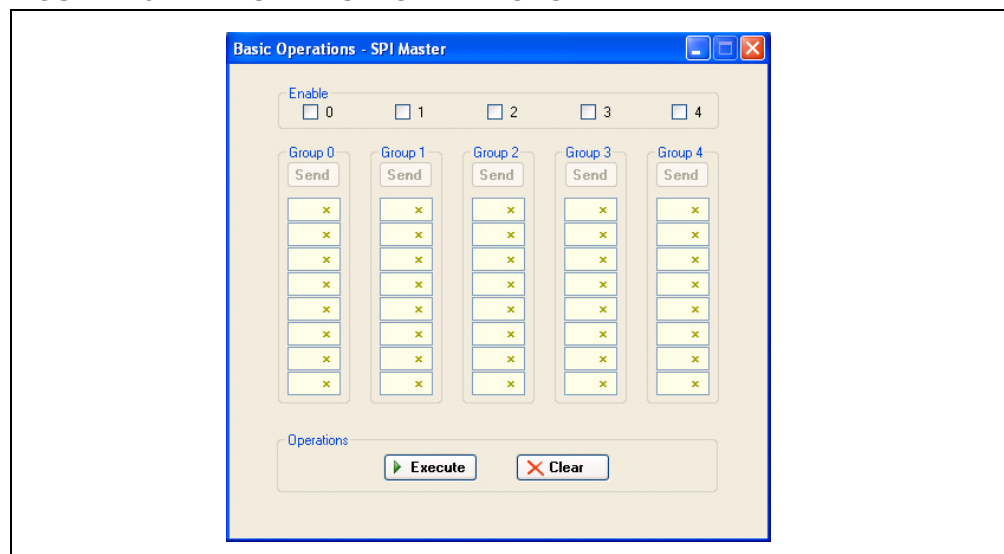
Clicking on the **Execute** button will execute the enabled columns in order from left to right.

The **Clear** button clears all boxes.

Note: The “x” indicates the value is a hexadecimal number. Clicking on “x” will toggle it to a “d” indicating that the value is a decimal number.

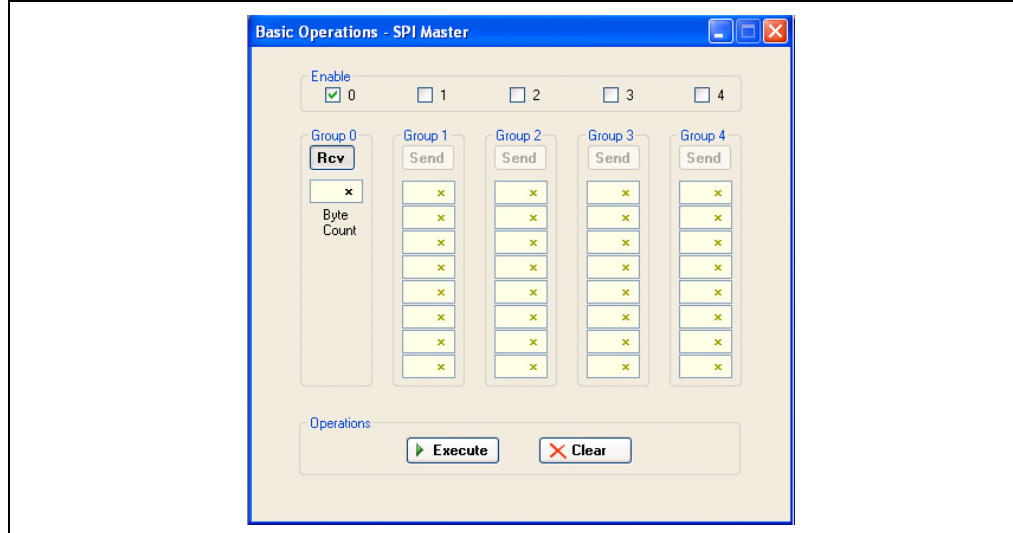
The commands will be logged in the Transactions window. A listing of the command abbreviations is given in Table 7-2.

FIGURE 7-3: SPI BASIC OPERATIONS



SPI and Microwire Master Communications

FIGURE 7-4: SPI BASIC OPERATIONS

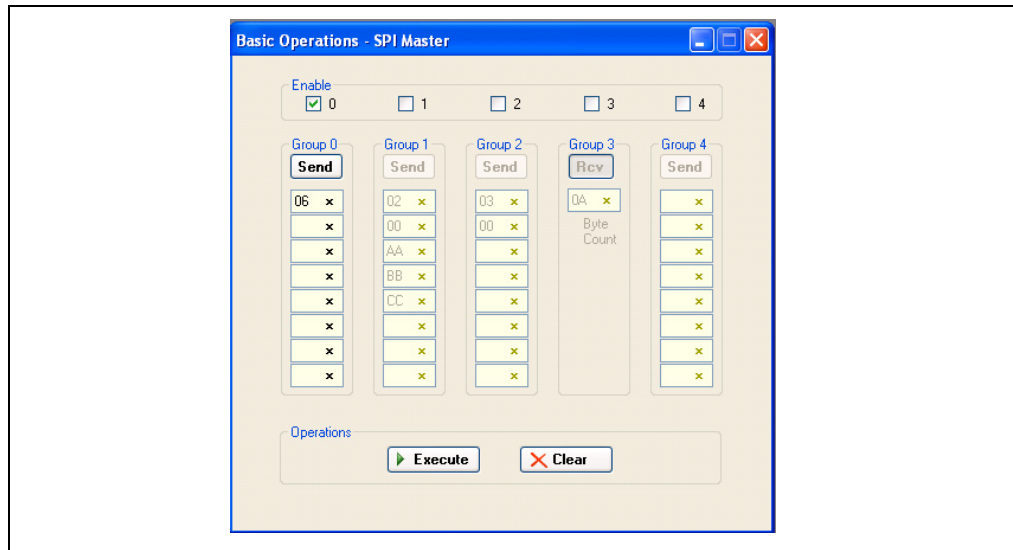


7.6.1 Basic Communications – Serial EEPROM Example

Figures 7-5 through 7-7 demonstrates how to communicate with a 25LC020A SPI serial EEPROM. Refer to the 25LC020A Data Sheet (DS21833) for a detailed explanation of its SPI communications.

Before data can be written to the 25LC020A, the write enable (WREN) latch must be set. This requires that CS be enabled, command 0x06 transmitted, and CS disabled. Figure 7-5 shows only Group 0 enabled. All other groups are disabled. Clicking on the **Execute** button will transmit only the WREN command. The command will be logged in the Transactions window.

FIGURE 7-5: SEEPROM EXAMPLE – WREN COMMAND



Once the WREN latch has been enabled, data can be written to the 25LC020A. Figure 7-6 shows that Group 0 has been disabled, and Group 1 enabled. Clicking on the **Execute** button will send the Write command (0x02), the memory address (0x00), followed by three bytes of data: 0xAA, 0xBB, and 0xCC. The command will be logged in the Transactions window.

FIGURE 7-6: SEEPROM EXAMPLE – WRITE BYTES

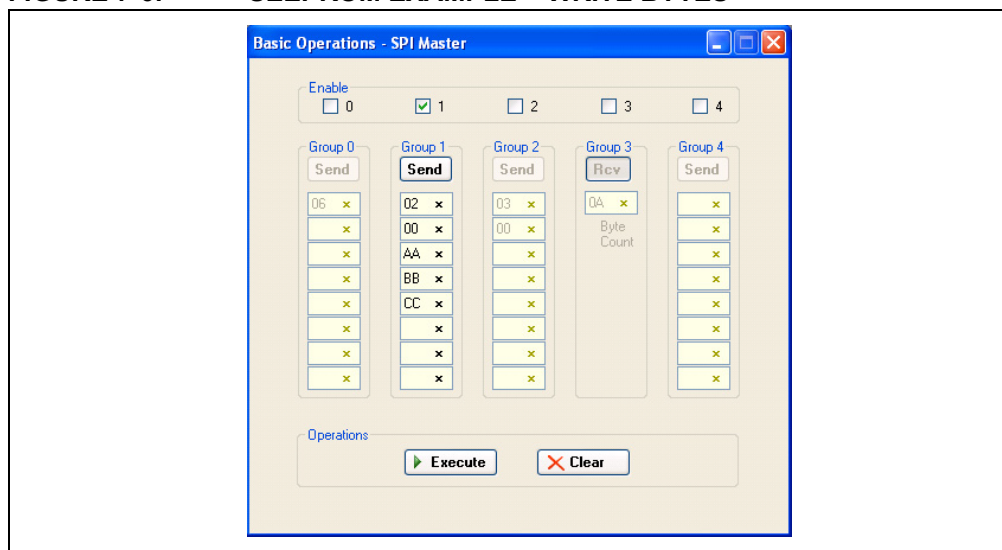
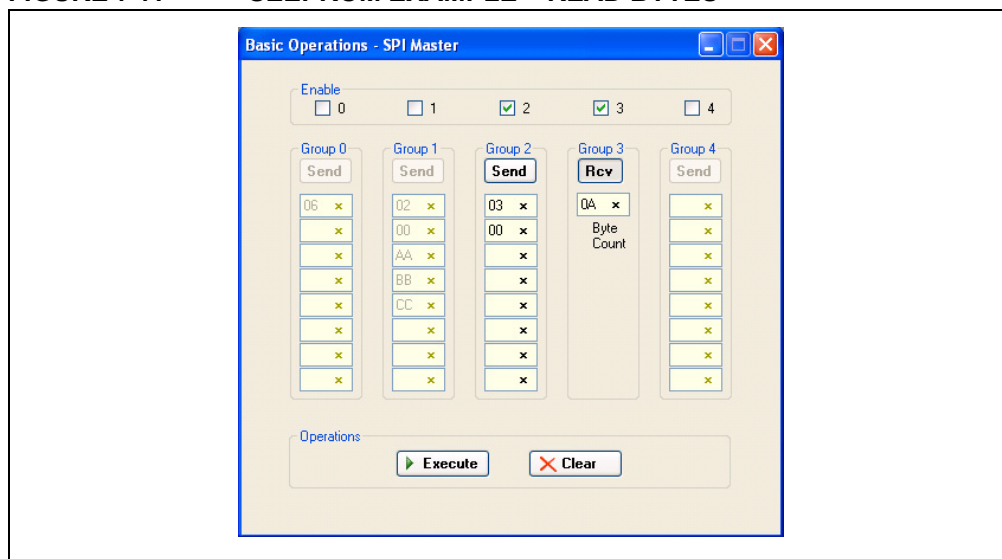


Figure 7-7 shows how to read data from the 25LC020A. Groups 0 and 1 are disabled, and Groups 2 and 3 are enabled. This example shows how data is transmitted and received in one transaction (Chip Select, \overline{CS} , active) between Groups. Clicking on the **Execute** button will send the Read command (0x03) and memory address (0x00) of Group 2 followed by a Read Ten Bytes command in Group 3. The commands and received data are displayed in the transactions window.

FIGURE 7-7: SEEPROM EXAMPLE – READ BYTES



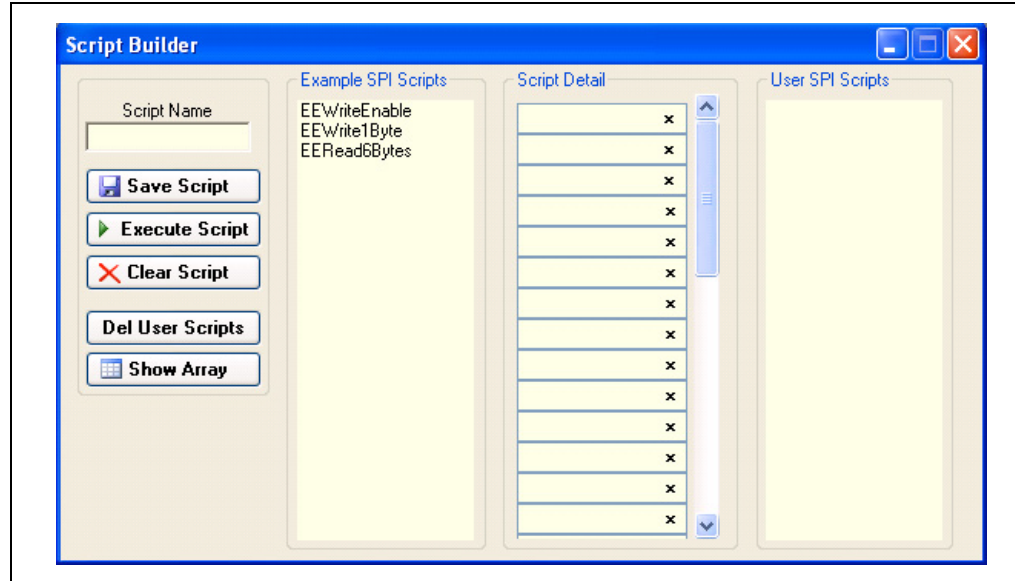
7.7 SCRIPT BUILDER

SPI commands can be combined into scripts, saved, and used over again. The Script Builder window is opened by selecting *Communications > Script > Script Builder* from the menu bar. The Script Builder is shown in Figure 7-8.

The Script Builder window is divided into four columns as shown in Figures 7-9 through 7-12.

SPI and Microwire Master Communications

FIGURE 7-8: SPI SCRIPT BUILDER

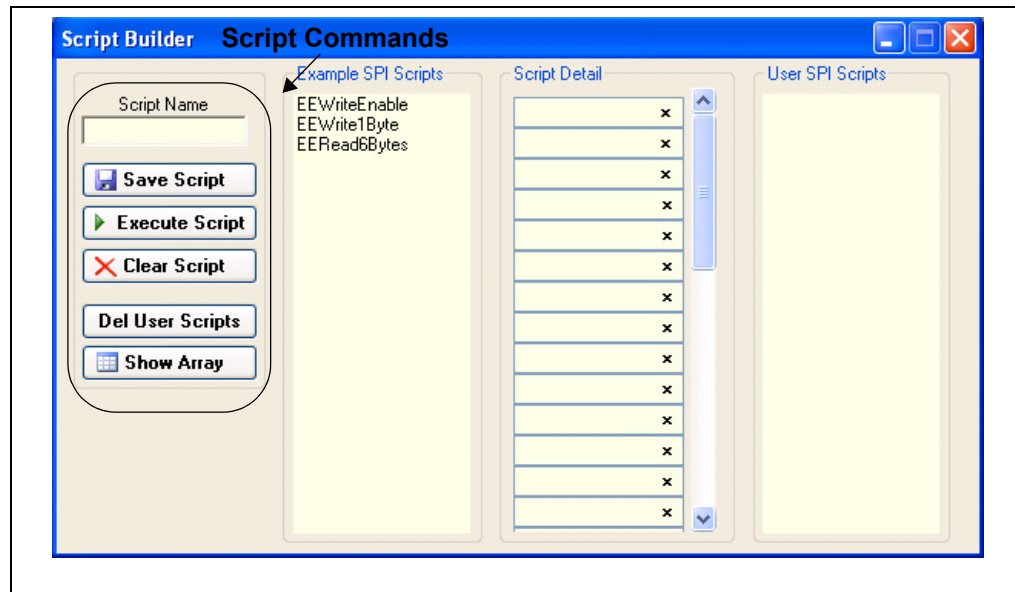


7.7.1 Script Commands

The left most column contains the Script Commands as shown in Figure 7-9.

- Script Name – Enter the name of the script
- Save Script – Saves the script
- Execute Script – Executes (performs) the script displayed in the Script Detail column
- Clear Script – Clears the Script Detail column
- Del User Scripts – Deletes scripts from the User Scripts column.
- Show Array – Displays a spreadsheet-like table in which large amounts of data may be entered. This data can be included in the script by right clicking in a Script Detail cell and choosing “Insert Array”.

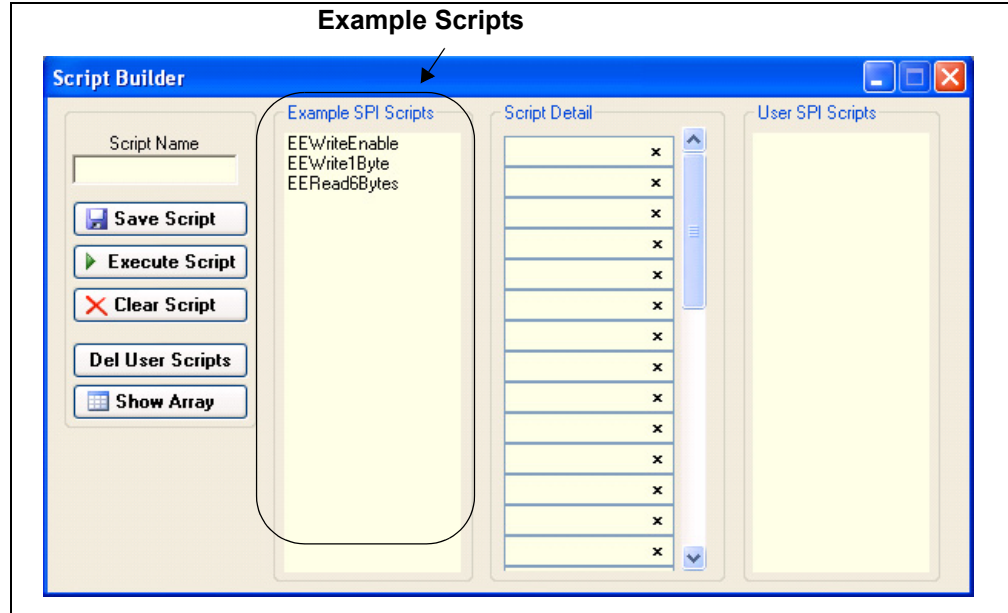
FIGURE 7-9: SPI SCRIPT BUILDER – SCRIPT COMMANDS



7.7.2 Example Scripts

The second column contains Example Scripts as shown in Figure 7-10. These can be studied to learn how to create or to edit custom scripts. To load the example script into the Script Detail column, either double click or right click and select from the local menu.

FIGURE 7-10: SPI SCRIPT BUILDER – EXAMPLE SCRIPTS



7.7.3 Script Detail

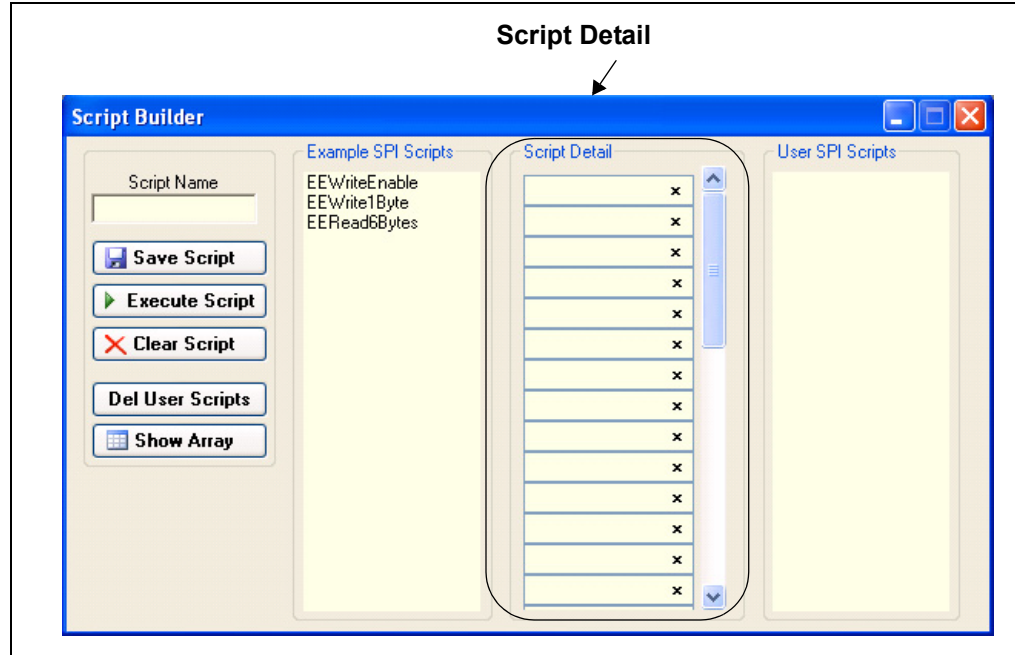
The third column contains Script Detail as shown in Figure 7-11. This column is used to create the script or view an existing script. More information about creating a customer script is discussed in **Section 7.7.5 “Creating A Script”**.

To load a user script from the User Scripts column into the Script Detail column, the user can double click or right click and select from the local menu.

Note: The “x” indicates the value is a hexadecimal number. Clicking on “x” will toggle it to a “d” indicating that the value is a decimal number.

SPI and Microwire Master Communications

FIGURE 7-11: SPI SCRIPT BUILDER – SCRIPT DETAIL



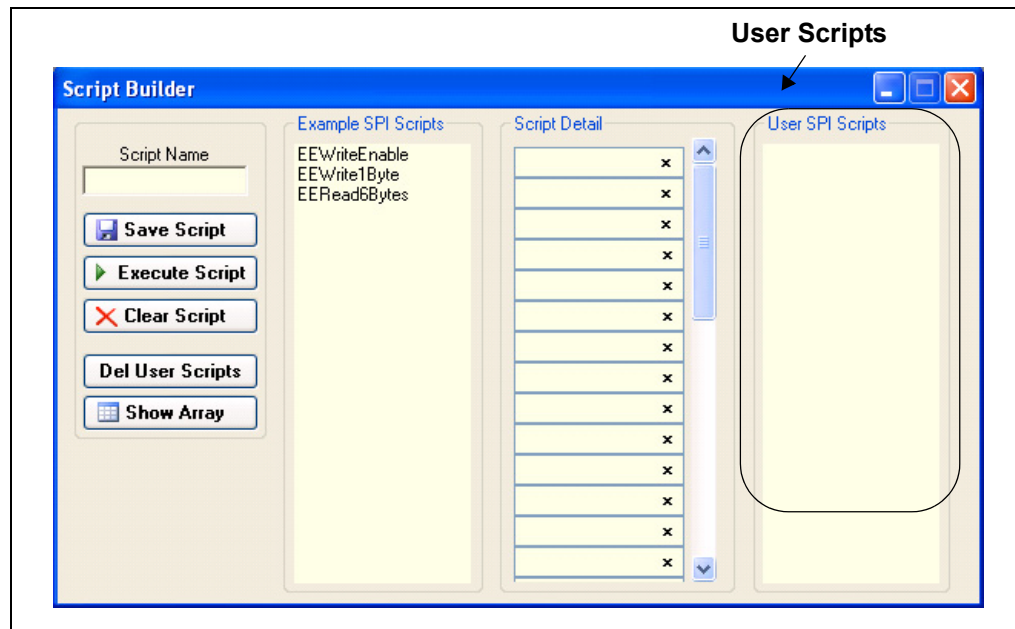
7.7.4 User Scripts

The fourth column contains User Scripts as shown in Figure 7-12. User scripts that are created, named, and saved are displayed in the User Scripts column.

To load a user script from the User Scripts column into the Script Detail column, the user can double click or right click and select from the local menu.

User Scripts can be deleted by right clicking and selecting Delete Script from the local menu.

FIGURE 7-12: SPI SCRIPT BUILDER – USER SCRIPTS



7.7.5 Creating A Script

Scripts are created by placing the cursor into the Script Detail column and right clicking. A local menu will be displayed as shown in Figure 7-13. Select from the choice of commands or script macro commands.

The sequence of macro commands are executed from top to bottom. Macro commands are entered by right clicking in the box and selecting from the local menu as shown in Figure 7-13.

Macro commands are entered according to the sequence of events as defined by the SPI bus protocol. Studying the example scripts is a good way to learn the sequence of events. The example scripts can also be modified and saved under a different name.

CAUTION

The choice of macro commands is very flexible. Therefore, the correctness of the script has to be verified by the user. The PICkit Serial Analyzer program does not verify the correctness of the script.

A complete listing of the available macro commands is given in Table 7-2. The macro command abbreviation will be displayed in the Transactions Window. The Transactions window keeps a running log of the commands and data sent to and from the target device.

FIGURE 7-13: SPI SCRIPT BUILDER – CREATING A SCRIPT

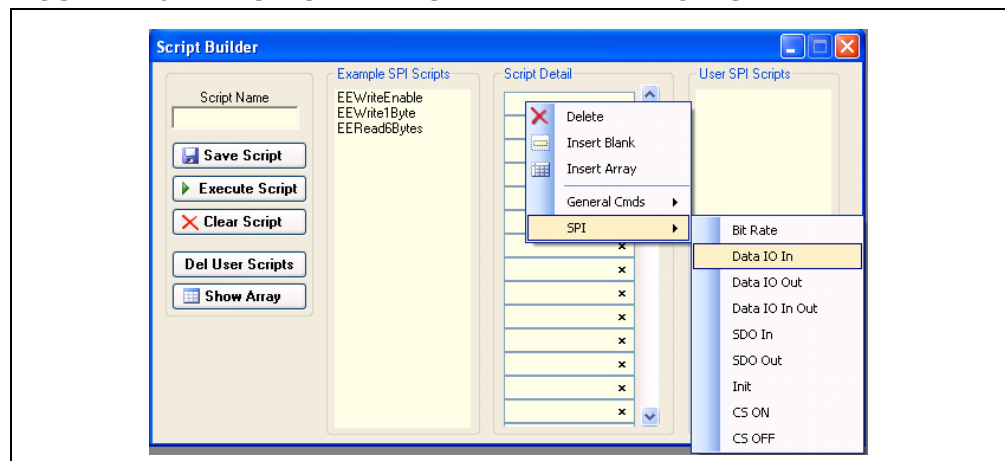


TABLE 7-2: SPI SCRIPT MACRO COMMAND

Macro Command	Command Abbreviation	Description
SPIBITRATE	[BR]	Set Bit Rate. Next byte is the scaler, followed by the pre-scaler.
SPIDATIN	[DI]	Input data. Next byte is the byte count.
SPIDATOUT	[DO]	Output data. Next byte is the byte count, followed by the data.
SPIDATIO	[DIO]	Simultaneous data in and out. Next byte is byte count, followed by the data.
SPISDOIN	[SI]	Set SDO pin to Input (tri-state)
SPISDOOUT	[SO]	Set SDO pin to Output
SPIINIT	[I]	Initialize SPI controller
SPICSON	[CSON]	Assert CS (active-low)

SPI and Microwire Master Communications

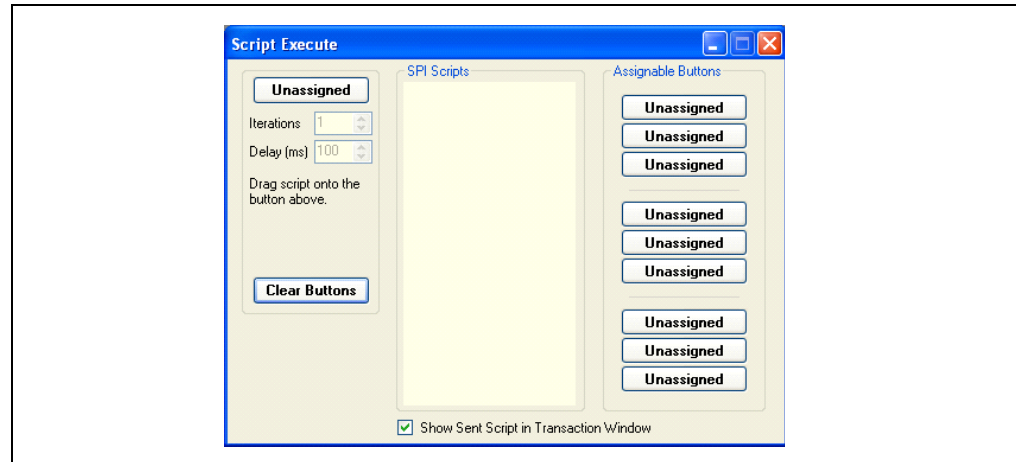
TABLE 7-2: SPI SCRIPT MACRO COMMAND (CONTINUED)

SPICSOFF	[CSOF]	De-assert \overline{CS} (active-low)
----------	--------	--

7.8 SCRIPT EXECUTE

The Script Execute window is shown in Figure 7-14. Once scripts are created using the Script Builder, they can be assigned to buttons in the Script Execute window. This makes a convenient window to execute multiple scripts either individually or iteratively. Script executing will be logged in the Transactions window.

FIGURE 7-14: SPI SCRIPT EXECUTE



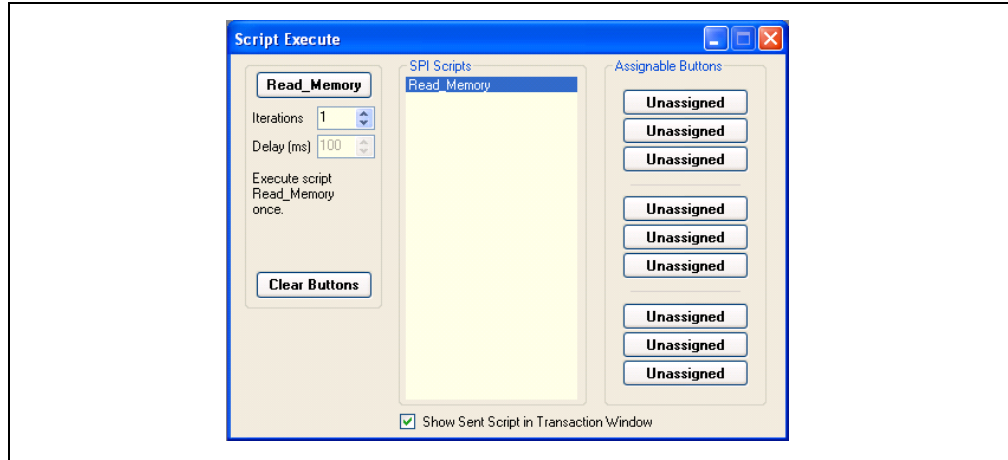
7.8.1 Assignable Buttons

User created scripts will be displayed in the central SPI Scripts column. To assign a script to a button, click on the script name and drag it to the desired Assignable Buttons in the right column. The script will be executed once each time the button is clicked. The Assignable Buttons can be cleared by clicking on the **Clear Buttons** button.

7.8.2 Iteration

Scripts can be executed a user defined number of times at a specified interval of time. Figure 7-15 shows an example. A script named Read_Memory has been assigned to the **Iteration** button in the left column. The number of iterations are entered in the Iterations box and the delay in millisecond in the Delay box. A summary of the iterations is displayed in the left column. The macro is executed when the **Iteration** button is clicked.

FIGURE 7-15: SPI SCRIPT EXECUTE – EXAMPLE



SPI and Microwire Master Communications

NOTES:

PICKit™ Serial Analyzer User's Guide

Chapter 8. USART Asynchronous Communications

8.1 INTRODUCTION

This chapter describes the USART Asynchronous Communications mode. USART Asynchronous data and commands can be entered using a Basic Communications window or by creating Script Commands.

It is assumed that the user is familiar with the USART Asynchronous protocol. For more information see:

- USART, AUSART, or EUSART chapter of the PIC microcontroller data sheet of interest
- A USART Asynchronous Communications tutorial is available on the Microchip Technology web site. Click on the links: Support -> Getting Started -> PIC MCU Tutorials -> USART - Using in Asynchronous Mode
- Several application notes are available on the Microchip Technology web site. Click on links: Design -> App Notes -> Function: Communication -> USART

8.2 HIGHLIGHTS

This chapter discusses:

- PICKit Serial Pin Assignments
- Selecting Communications Mode
- Configuring USART Asynchronous Communications Mode
- Communications: Basic Operations
- Script Builder
- Script Execute

8.3 PICKit SERIAL PIN ASSIGNMENTS

The PICKit Serial Analyzer pin assignments for USART Asynchronous Communications mode are:

TABLE 8-1: USART ASYNCHRONOUS PIN ASSIGNMENTS

Pin	Label	Type	Description
1	TX	Output	Transmit Data (with respect to the PICKit™ Serial Analyzer)
2	+V	Power	Target Power
3	GND	Power	Ground
4	AUX1	Input/Output	Auxiliary I/O port pin No. 1
5	AUX2	Input/Output	Auxiliary I/O port pin No. 2
6	RX	Input	Receive Data (with respect to the PICKit™ Serial Analyzer)

PICkit™ Serial Analyzer User's Guide

8.4 SELECTING COMMUNICATIONS MODE

The USART Asynchronous Communications mode is selected from the Configuration Wizard or menu bar.

Configuration Wizard – Select *PICkit Serial Analyzer > Run Configuration Wizard* from the menu bar

Menu Bar – Select *PICkit Serial Analyzer > Select Communications Mode > USART Asynchronous*

8.5 CONFIGURING USART ASYNCHRONOUS COMMUNICATIONS MODE

Once the communications mode has been selected, it is configured from the Configuration Wizard or menu bar.

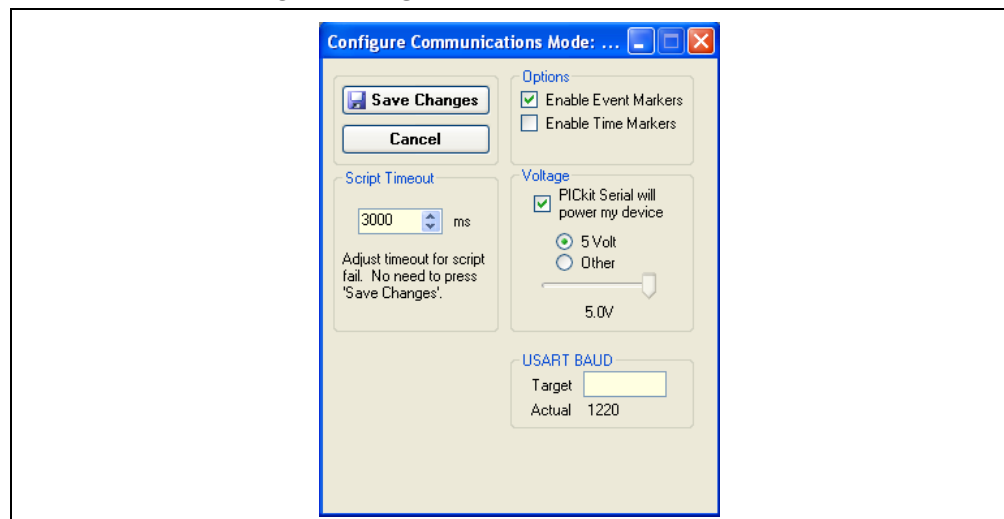
Configuration Wizard – Select *PICkit Serial Analyzer > Run Configuration Wizard* from the menu bar

Menu Bar – Select *PICkit Serial Analyzer > Configure Communications Mode*

The Configure Mode window will open. Depending on the view selected, the Basic View (Figure 8-1) displays a minimum choice of configurations commands. In the Advanced View (Figure 8-2) displays an extended choice of configuration commands.

Save the configuration by clicking on the **Save Changes** button.

FIGURE 8-1: USART ASYNCHRONOUS CONFIGURE COMMUNICATIONS MODE – BASIC VIEW



OPTIONS

- Enable Event Markers – Enable event markers
- Enable Time Markers – Enable 'time' stamp to accompany all event markers

VOLTAGE

- PICkit Serial will power my device – Select the check box if the PICkit Serial will power the target device. The target can be powered at 5 VDC or a user selectable variable voltage.

CAUTION

Even though the voltage can be set as low as 0 VDC, it is up to the user to verify the required operating voltage of the target device.

USART Asynchronous Communications

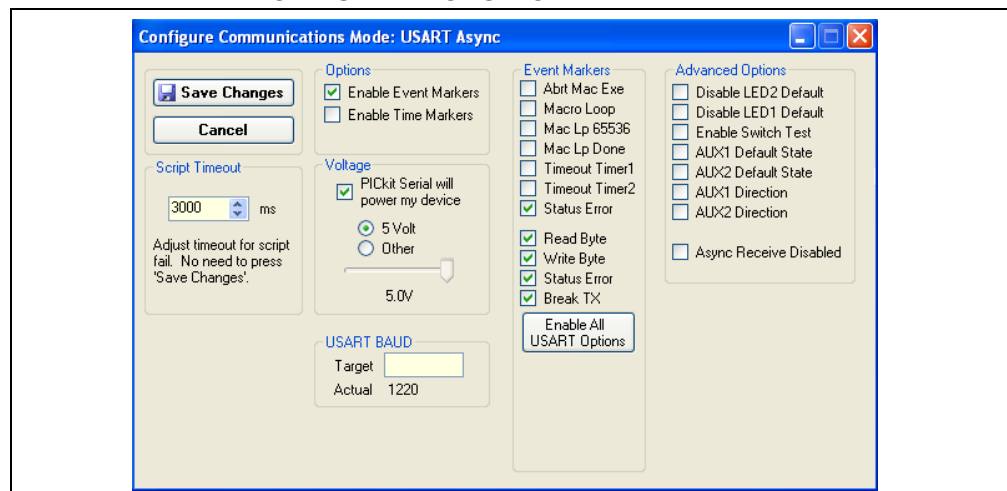
CAUTION

The USB port current limit is set to 100 mA. If the target plus PICKit Serial Analyzer exceeds this current limit, the USB port will turn off. The target may be powered externally if more power is required.

USART BAUD

Enter the desired USART symbol rate (Baud) in the text box.

FIGURE 8-2: USART ASYNCHRONOUS CONFIGURE COMMUNICATIONS MODE – ADVANCED VIEW



SCRIPT TIMEOUT

When sending scripts, the software will wait a maximum of Script Timeout ms to receive a script complete tag before issuing an error. If your hardware is slow to respond or you are transferring a lot of data, you may need to increase the Script Timeout to avoid errors.

EVENT MARKERS

- Abt Mac Exe – Enable event marker: abort 'macro' execution
- Macro Loop – Enable event marker: top of 'macro' loop
- Mac Lp 65536 – Enable event marker: 'macro' loop count overflow (i.e., 65536)
- Mac Lp Done – Enable event marker: 'macro' loop iterations complete
- Timeout Timer1 – Enable event marker: Timer1 expired
- Timeout Timer2 – Enable event marker: Timer2 expired
- Status Error – Enable event marker: change in status byte
- Read Byte – Enable event marker – read byte
- Write Byte – Enable event marker – write byte
- Status Error – Enable event marker – change in USART status byte
- Break TX – Enable event marker – A "Break" has been transmitted

ADVANCED OPTIONS

- Disable LED2 Default – Disable default LED2 behavior (LED2 = Yellow 'Target' LED)
- Disable LED1 Default – Disable default LED1 behavior (LED1 = Red 'Busy' LED)
- Enable Switch Test – Enable low level switch test:

PICKit™ Serial Analyzer User's Guide

- AUX1 Default State – AUX1 communication line – default state (0 | 1)
- AUX2 Default State – AUX2 communication line – default state (0 | 1)
- AUX1 Direction – AUX1 communication line – direction: 1: input, 0: output
- AUX2 Direction – AUX2 communication line – direction: 1: input, 0: output
- Async Receive Disabled – Received Data is disabled

8.6 COMMUNICATIONS: BASIC OPERATIONS

The USART Asynchronous Operations window can be opened by selecting:

- Communications: Basic Operations from the tool bar, or
- Communications > Basic Operations from the menu bar

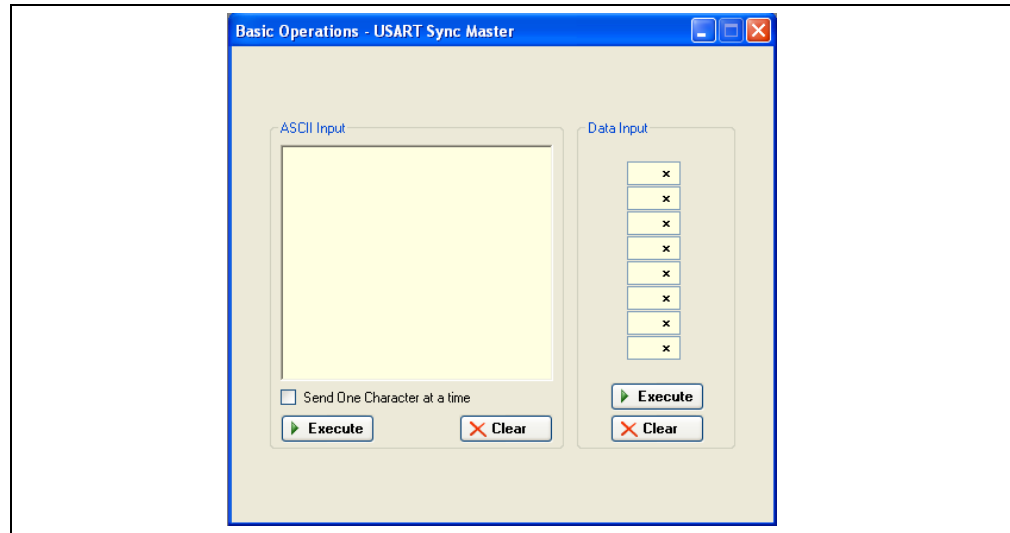
The USART Asynchronous Basic Operations window is shown in Figure 8-3. There are two basic communications commands, Read and Write.

Data can be transmitted to the target device as 7-bit ASCII or 8-bit byte. 7-bit ASCII data is entered in the left hand window. 8-bit byte is entered in the right hand column. Both transmitted and received data is displayed in the Transaction window.

Note: The “x” indicates the value is a hexadecimal number. Clicking on “x” will toggle it to a “d” indicating that the value is a decimal number.

The command will be logged in the Transactions window. A listing of the command abbreviations is given in Table 8-2.

FIGURE 8-3: USART ASYNCHRONOUS BASIC OPERATIONS



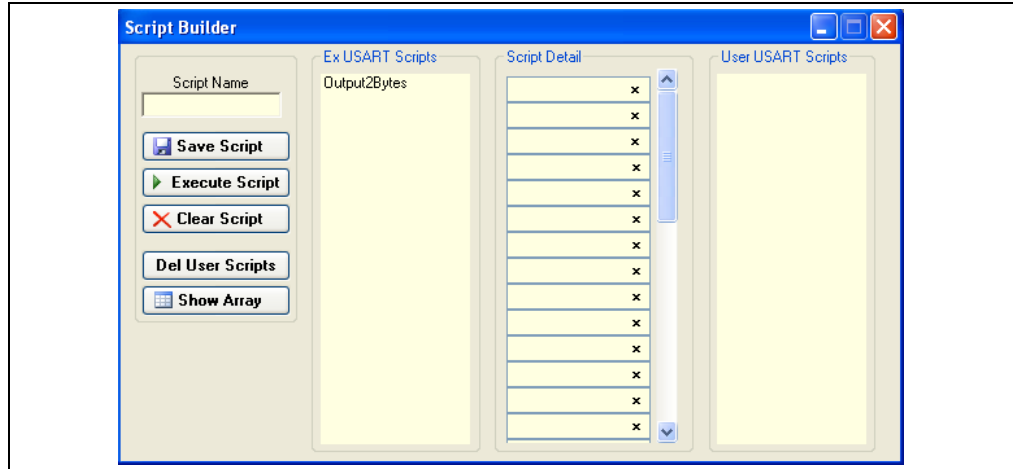
8.7 SCRIPT BUILDER

USART Asynchronous commands can be combined into scripts, saved, and used over again. The Script Builder window is opened by selecting Communications > Script > Script Builder from the menu bar. The Script Builder is shown in Figure 8-4.

The Script Builder window is divided into four columns as shown in Figures 8-5 through 8-8.

USART Asynchronous Communications

FIGURE 8-4: USART ASYNCHRONOUS SCRIPT BUILDER

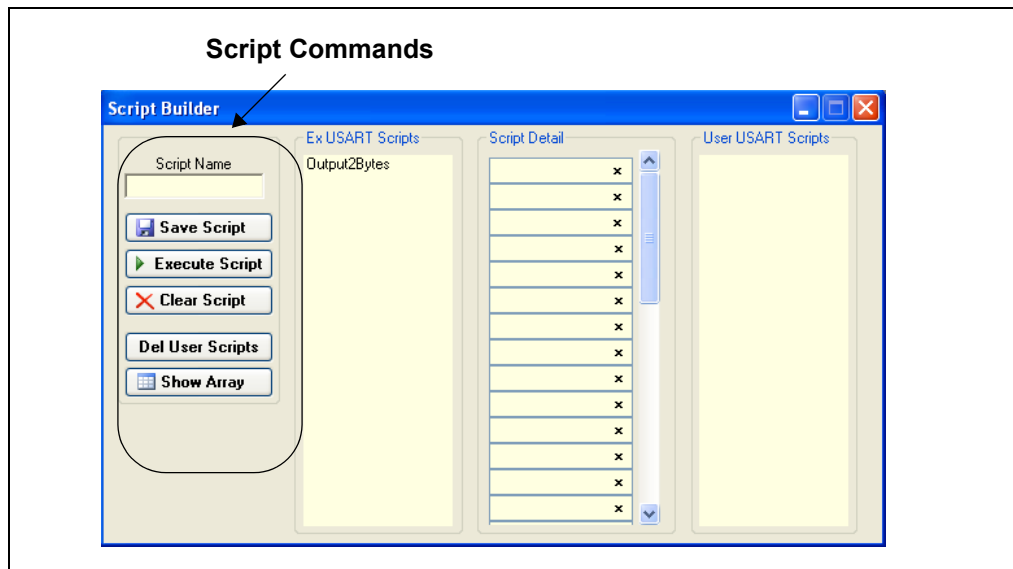


8.7.1 Script Commands

The left most column contains the Script Commands as shown in Figure 8-5.

- Script Name – Enter the name of the script
- Save Script – Saves the script
- Execute Script – Executes (performs) the script displayed in the Script Detail column
- Clear Script – Clears the Script Detail column
- Del User Scripts – Deletes scripts from the User Scripts column.
- Show Array – Displays a spreadsheet-like table in which large amounts of data may be entered. This data can be included in the script by right clicking in a Script Detail cell and choosing "Insert Array".

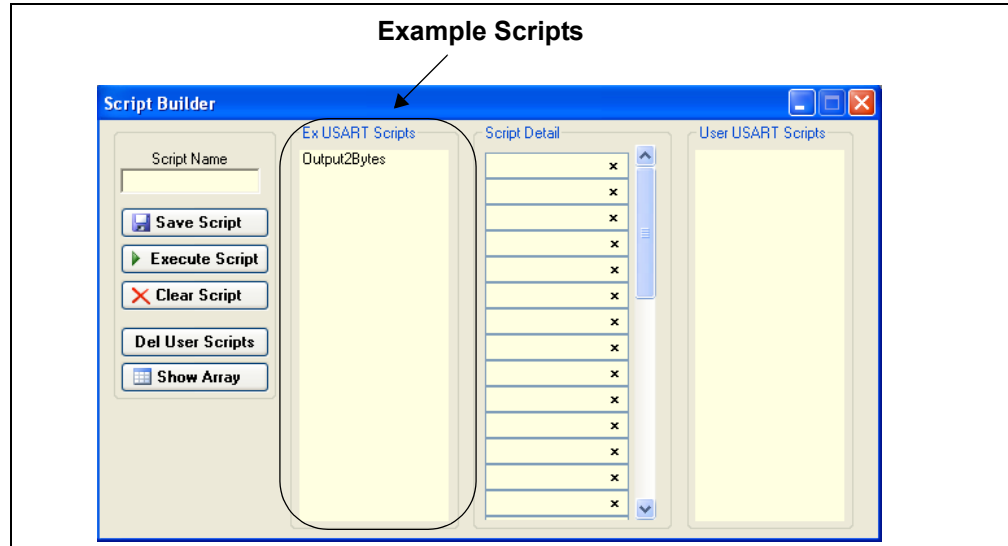
FIGURE 8-5: USART ASYNCHRONOUS SCRIPT BUILDER – SCRIPT COMMANDS



8.7.2 Example Scripts

The second column contains Example Scripts as shown in Figure 8-6. These can be studied to learn how to create or to edit custom scripts. To load the example script into the Script Detail column, either double click or right click and select from the local menu.

FIGURE 8-6: USART ASYNCHRONOUS SCRIPT BUILDER – EXAMPLE SCRIPTS



8.7.3 Script Detail

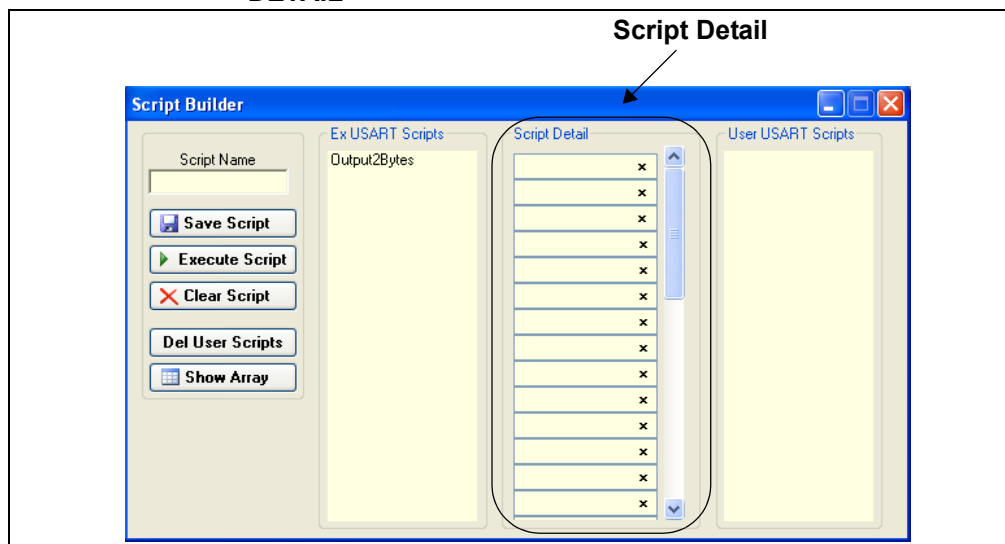
The third column contains Script Detail as shown in Figure 8-7. This column is used to create the script or view an existing script. More information about creating a customer script is discussed in **Section 8.7.5 “Creating A Script”**.

To load a user script from the User Scripts column into the Script Detail column, the user can double click or right click and select from the local menu.

Note: The “x” indicates the value is a hexadecimal number. Clicking on “x” will toggle it to a “d” indicating that the value is a decimal number.

USART Asynchronous Communications

FIGURE 8-7: USART ASYNCHRONOUS SCRIPT BUILDER – SCRIPT DETAIL



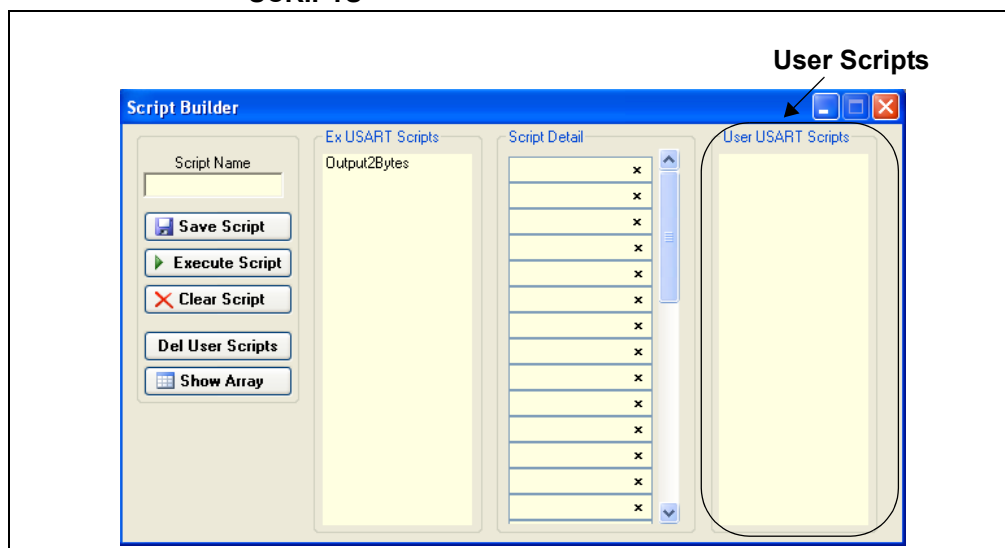
8.7.4 User Scripts

The fourth column contains User Scripts as shown in Figure 8-8. User scripts that are created, named, and saved are displayed in the User Scripts column.

To load a user script from the User Scripts column into the Script Detail column, the user can double click or right click and select from the local menu.

User Scripts can be deleted by right clicking and selecting Delete Script from the local menu.

FIGURE 8-8: USART ASYNCHRONOUS SCRIPT BUILDER – USER SCRIPTS



8.7.5 Creating A Script

Scripts are created by placing the cursor into the Script Detail column and right clicking. A local menu will be displayed as shown in Figure 8-9. Select from the choice of commands or script macro commands.

PICKit™ Serial Analyzer User's Guide

The sequence of macro commands are executed from top to bottom. Macro commands are entered by right clicking in the box and selecting from the local menu as shown in Figure 8-9.

Macro commands are entered according to the sequence of events as defined by the USART Asynchronous protocol. Studying the example scripts is a good way to learn the sequence of events. The example scripts can also be modified and saved under a different name.

CAUTION

The choice of macro commands is very flexible. Therefore, the correctness of the script has to be verified by the user. The PICKit Serial Analyzer program does not verify the correctness of the script.

A complete listing of the available macro commands is given in Table 8-2. The macro command abbreviation will be displayed in the Transactions Window. The Transactions window keeps a running log of the commands and data sent to and from the target device.

FIGURE 8-9: USART ASYNCHRONOUS SCRIPT BUILDER – CREATING A SCRIPT

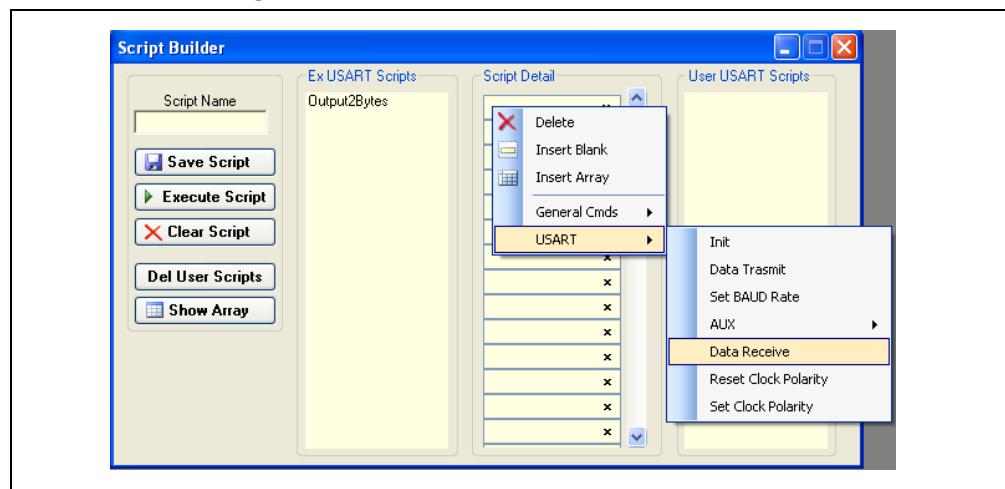


TABLE 8-2: USART SCRIPT MACRO COMMANDS

Macro Command	Command Abbreviation	Description
USDATATX	[TX]	Transmit data. Next byte is the byte count, followed by the data.
USDATARX	[RX]	Receive data. Next byte is the byte count.
USDATARXEN	[ER]	Enable Receive data monitor
USDATARXDS	[DR]	Disable Receive data monitor
USBREAKTX	[BK]	Send Break
USBRKDATTX	[BKD]	Send Break, then data byte. Next byte is the data byte.
USBAUD	[BD]	Set BAUD Rate. Next byte is BAUD (LSB) followed by BAUD (MSB).
US9BITSET	[9S]	Set 9-bit Data mode
US9BITRST	[9R]	Reset 9-bit Data mode (sets 8-bit)
USCLKPOLSET	[CS]	Set CLOCK POLARITY bit

USART Asynchronous Communications

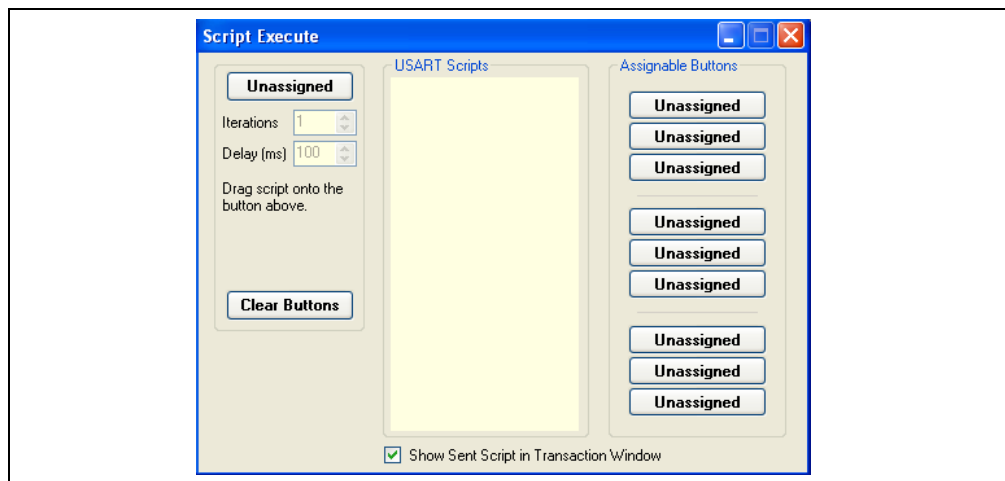
TABLE 8-2: USART SCRIPT MACRO COMMANDS (CONTINUED)

USCLKPOLRST	[CR]	Reset CLOCK POLARITY bit
USINIT	[I_]	Initialize USART controller
USRESET	[RE]	Reset USART controller.
USAUX1RST	[A1RST]	Reset Aux1
USAUX1SET	[A1RST]	Set Aux1
USAUX1OUT	[A1OUT]	Set Aux1 direction to Output
USAUX1IN	[A1IN]	Set Aux1 direction to Input
USAUX1W0	[A1W0]	Aux1 Wait 0
USAUX1W1	[A1W1]	Aux1 Wait 1
USAUX2RST	[A2RST]	Reset Aux2
USAUX2SET	[A2RST]	Set Aux2
USAUX2OUT	[A2OUT]	Set Aux2 direction to Output
USAUX2IN	[A2IN]	Set Aux2 direction to Input
USAUX2W0	[A2W0]	Aux2 Wait 0
USAUX2W1	[A2W1]	Aux2 Wait 1

8.8 SCRIPT EXECUTE

The Script Execute window is shown in Figure 8-10. Once scripts are created using the Script Builder, they can be assigned to buttons in the Script Execute window. This makes a convenient window to execute multiple scripts either individually or iteratively. Script executing will be logged in the Transactions window.

FIGURE 8-10: USART ASYNCHRONOUS SCRIPT EXECUTE



8.8.1 Assignable Buttons

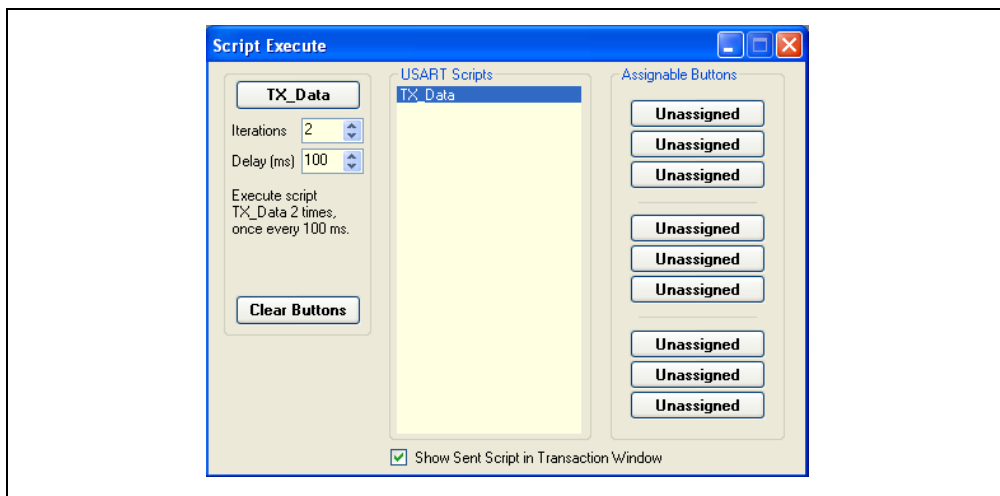
User created scripts will be displayed in the central USART Scripts column. To assign a script to a button, click on the script name and drag it to the desired Assignable Buttons in the right column. The script will be executed once each time the button is clicked.

The Assignable Buttons can be cleared by clicking on the **Clear Buttons** button.

8.8.2 Iteration

Scripts can be executed a user defined number of times at a specified interval of time. Figure 8-11 shows an example. A script named TX_Data has been assigned to the **Iteration** button in the left column. The number of iterations is entered in the Iterations box and the delay in milliseconds in the Delay box. A summary of the iterations is displayed in the left column. The macro is executed when the **Iteration** button is clicked.

FIGURE 8-11: USART ASYNCHRONOUS SCRIPT EXECUTE – EXAMPLE



Chapter 9. USART Master Synchronous Communications

9.1 INTRODUCTION

This chapter describes the USART Synchronous Master Communications mode. USART Synchronous Master data and commands can be entered using a Basic Communications window or by creating Script Commands.

It is assumed that the user is familiar with the USART Synchronous protocol. For more information see:

- USART, AUSART, or EUSART chapter of the PIC microcontroller data sheet of interest

9.2 HIGHLIGHTS

This chapter discusses:

- PICKit Serial Pin Assignments
- Selecting Communications Mode
- Configuring USART Synchronous Master Communications Mode
- Communications: Basic Operations
- Script Builder
- Script Execute

9.3 PICKit SERIAL PIN ASSIGNMENTS

The PICKit Serial Analyzer pin assignments for USART Synchronous Master Communications mode are:

TABLE 9-1: USART SYNCHRONOUS MASTER PIN ASSIGNMENTS

Pin	Label	Type	Description
1	CK	Output	Clock
2	+V	Power	Target Power
3	GND	Power	Ground
4	AUX1	Input/Output	Auxiliary I/O port pin No. 1
5	AUX2	Input/Output	Auxiliary I/O port pin No. 2
6	DT	Input/Output	Data

9.4 SELECTING COMMUNICATIONS MODE

The USART Synchronous Master Communications mode is selected from the Configuration Wizard or menu bar.

Configuration Wizard – Select *PICKit Serial Analyzer > Run Configuration Wizard* from the menu bar

Menu Bar – Select *PICKit Serial Analyzer > Select Communications Mode > USART Synchronous Master*

PICKit™ Serial Analyzer User's Guide

9.5 CONFIGURING USART SYNCHRONOUS MASTER COMMUNICATIONS MODE

Once the communications mode has been selected, it is configured from the Configuration Wizard or menu bar.

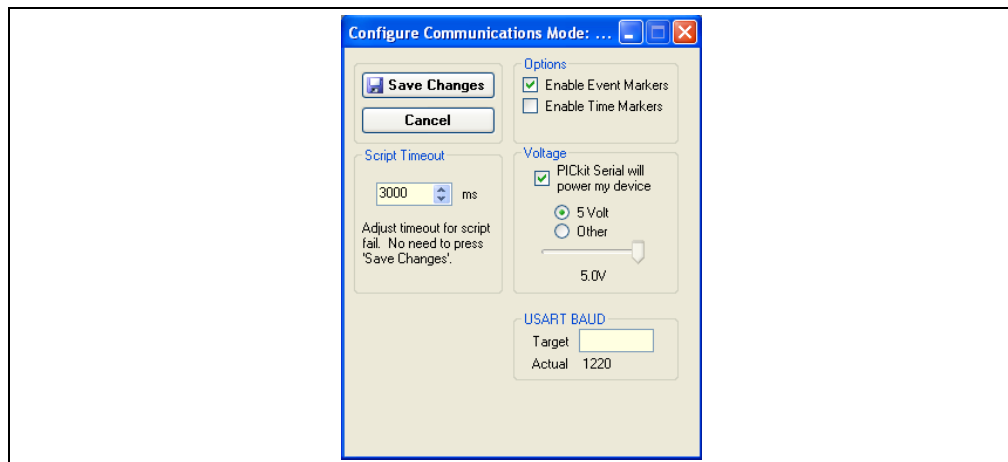
Configuration Wizard – Select *PICKit Serial Analyzer > Run Configuration Wizard* from the menu bar

Menu Bar – Select *PICKit Serial Analyzer > Configure Communications Mode*

The Configure Mode window will open. Depending on the view selected, the Basic View (Figure 9-1) displays a minimum choice of configurations commands. In the Advanced View (Figure 9-2) displays an extended choice of configuration commands.

Save the configuration by clicking on the **Save Changes** button.

FIGURE 9-1: USART SYNCHRONOUS MASTER CONFIGURE COMMUNICATIONS MODE – BASIC VIEW



OPTIONS

- Enable Event Markers – Enable event markers
- Enable Time Markers – Enable 'time' stamp to accompany all event markers

VOLTAGE

- PICKit Serial will power my device – Select the check box if the PICKit Serial will power the target device. The target can be powered at 5 VDC or a user selectable variable voltage.

CAUTION

Even though the voltage can be set as low as 0 VDC, it is up to the user to verify the required operating voltage of the target device.

CAUTION

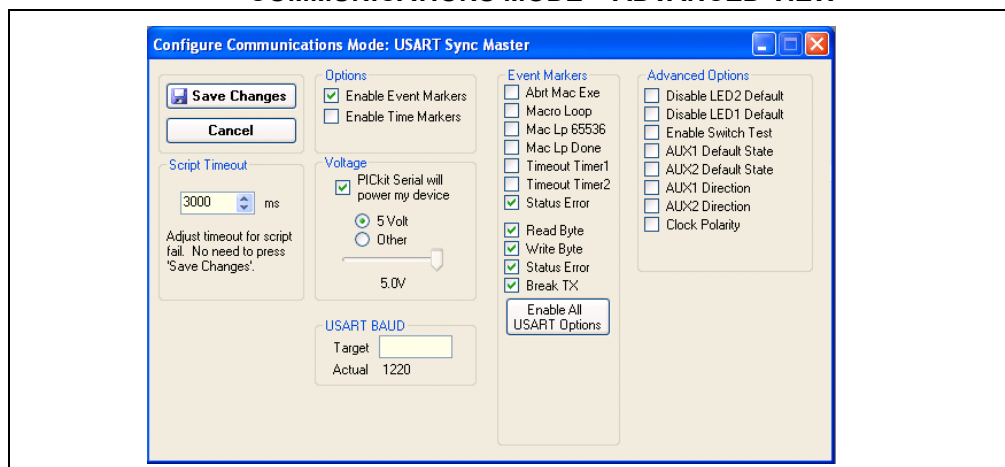
The USB port current limit is set to 100 mA. If the target plus PICKit Serial Analyzer exceeds this current limit, the USB port will turn off. The target may be powered externally if more power is required.

USART BAUD

Enter the desired USART symbol rate (Baud) in the text box.

USART Master Synchronous Communications

FIGURE 9-2: USART SYNCHRONOUS MASTER CONFIGURE COMMUNICATIONS MODE – ADVANCED VIEW



SCRIPT TIMEOUT

When sending scripts, the software will wait a maximum of Script Timeout ms to receive a script complete tag before issuing an error. If your hardware is slow to respond or you are transferring a lot of data, you may need to increase the Script Timeout to avoid errors.

EVENT MARKERS

- Abt Mac Exe – Enable event marker: abort ‘macro’ execution
- Macro Loop – Enable event marker: top of ‘macro’ loop
- Mac Lp 65536 – Enable event marker: ‘macro’ loop count overflow (i.e., 65536)
- Mac Lp Done – Enable event marker: ‘macro’ loop iterations complete
- Timeout Timer1 – Enable event marker: Timer1 expired
- Timeout Timer2 – Enable event marker: Timer2 expired
- Status Error – Enable event marker: change in status byte
- Read Byte – Enable event marker – read byte
- Write Byte – Enable event marker – write byte
- Status Error – Enable event marker – change in USART status byte
- Break TX – Enable event marker – A “Break” has been transmitted

ADVANCED OPTIONS

- Disable LED2 Default – Disable default LED2 behavior (LED2 = Yellow ‘Target’ LED)
- Disable LED1 Default – Disable default LED1 behavior (LED1 = Red ‘Busy’ LED)
- Enable Switch Test – Enable low level switch test:
 - Switch Off (not depressed) – blink LED1, LED2 off
 - Switch ON (depressed) – blink LED2, LED1 off
- AUX1 Default State – AUX1 communication line – default state (0 | 1)
- AUX2 Default State – AUX2 communication line – default state (0 | 1)
- AUX1 Direction – AUX1 communication line – direction: 1: input, 0: output
- AUX2 Direction – AUX2 communication line – direction: 1: input, 0: output
- Clock Polarity – Checked means the polarity is inverted, unchecked means it is not

9.6 COMMUNICATIONS: BASIC OPERATIONS

The USART Asynchronous Operations window can be opened by selecting:

- *Communications: Basic Operations* from the tool bar, or
- *Communications > Basic Operations* from the menu bar

The USART Synchronous Master Basic Operations window is shown in Figure 9-3. There are two basic communications commands, Read and Write.

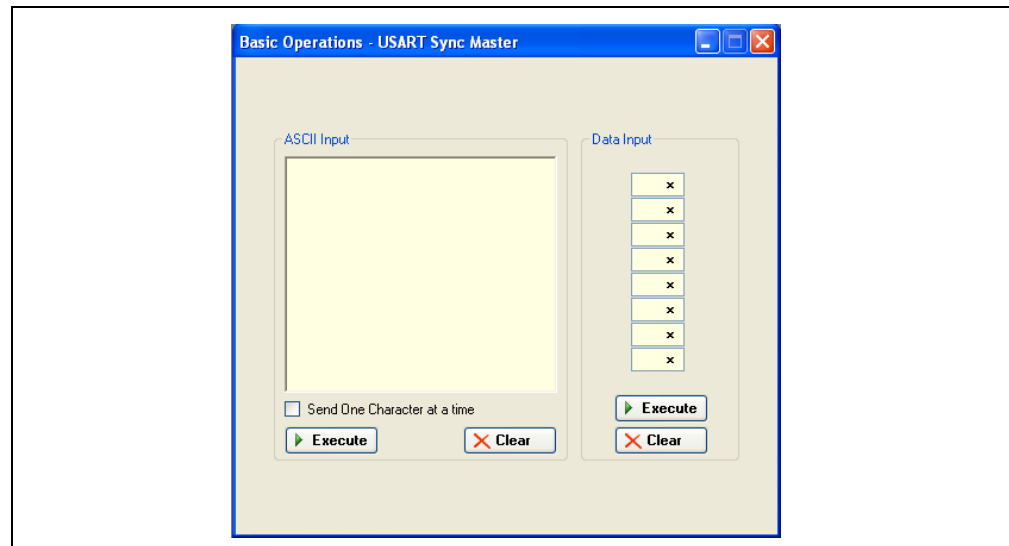
Data can be transmitted to the target device as 7-bit ASCII or 8-bit byte. 7-bit ASCII data is entered in the left hand window. 8-bit byte is entered in the right hand column.

Both transmitted and received data is displayed in the transaction window.

Note: The “x” indicates the value is a hexadecimal number. Clicking on “x” will toggle it to a “d” indicating that the value is a decimal number.

The command will be logged in the Transactions window. A listing of the command abbreviations is given in Table 9-2.

FIGURE 9-3: USART SYNCHRONOUS MASTER BASIC OPERATIONS



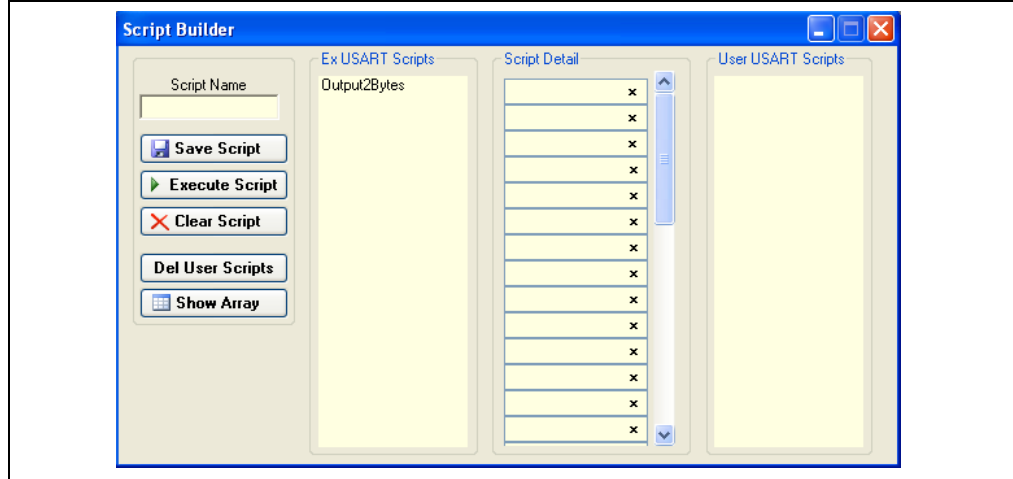
9.7 SCRIPT BUILDER

USART Asynchronous commands can be combined into scripts, saved, and used over again. The Script Builder window is opened by selecting *Communications > Script > Script Builder* from the menu bar. The Script Builder is shown in Figure 9-4.

The Script Builder window is divided into four columns as shown in Figures 9-5 through 9-8.

USART Master Synchronous Communications

FIGURE 9-4: USART SYNCHRONOUS MASTER SCRIPT BUILDER

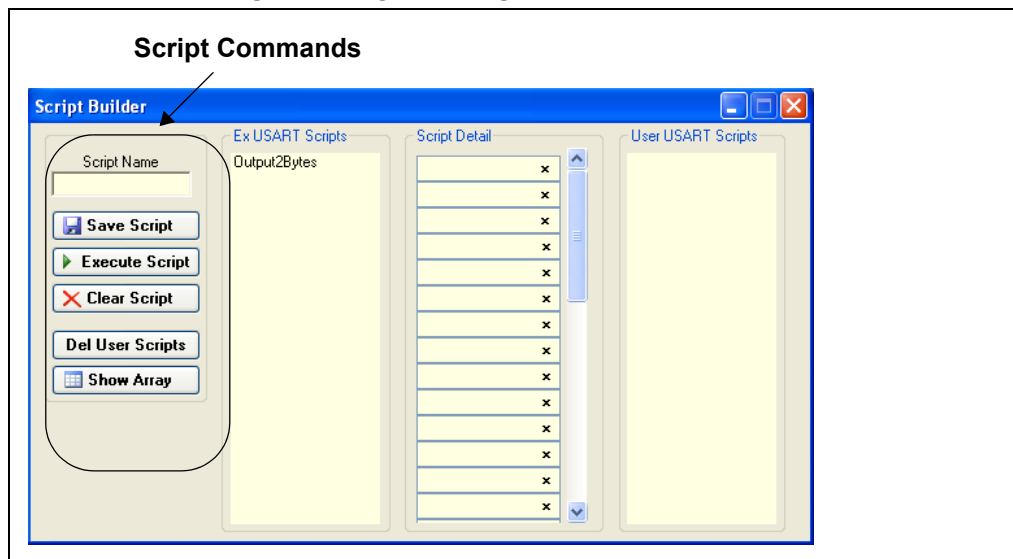


9.7.1 Script Commands

The left most column contains the Script Commands as shown in Figure 9-5.

- Script Name – Enter the name of the script
- Save Script – Saves the script
- Execute Script – Executes (performs) the script displayed in the Script Detail column
- Clear Script – Clears the Script Detail column
- Del User Scripts – Deletes scripts from the User Scripts column
- Show Array – Displays a spreadsheet-like table in which large amounts of data may be entered. This data can be included in the script by right clicking in a Script Detail cell and choosing "Insert Array".

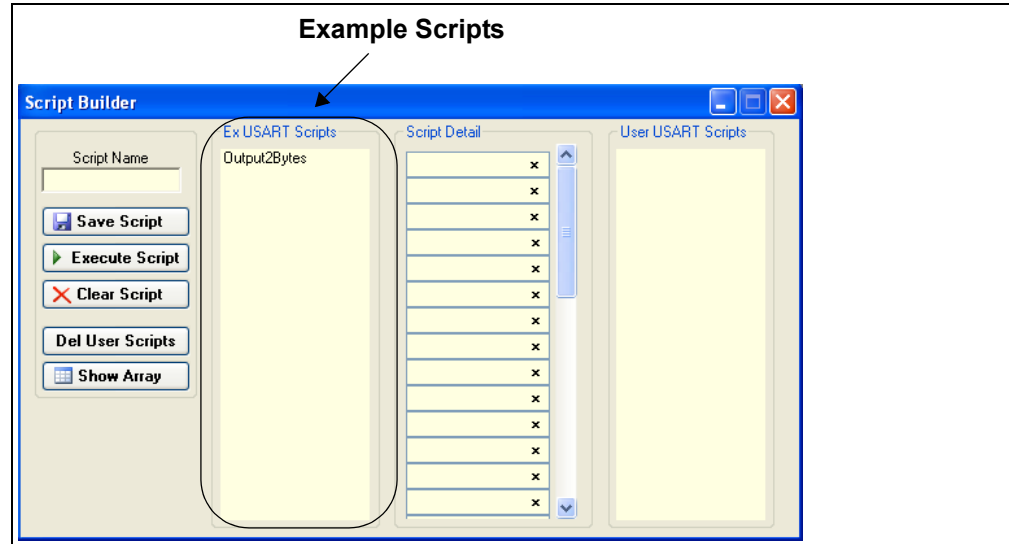
FIGURE 9-5: USART SYNCHRONOUS MASTER SCRIPT BUILDER – SCRIPT COMMANDS



9.7.2 Example Scripts

The second column contains Example Scripts as shown in Figure 9-6. These can be studied to learn how to create or to edit custom scripts. To load the example script into the Script Detail column, either double click or right click and select from the local menu.

FIGURE 9-6: USART SYNCHRONOUS MASTER SCRIPT BUILDER – EXAMPLE SCRIPTS



9.7.3 Script Detail

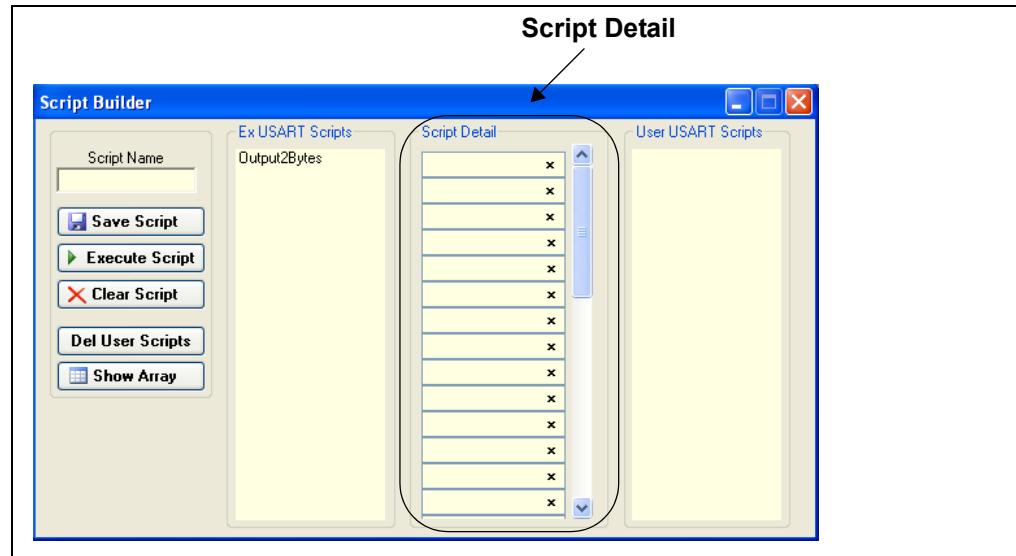
The third column contains Script Detail as shown in Figure 9-7. This column is used to create the script or view an existing script. More information about creating a customer script is discussed in **Section 9.7.5 “Creating a Script”**.

To load a user script from the User Scripts column into the Script Detail column, the user can double click or right click and select from the local menu.

Note: The “x” indicates the value is a hexadecimal number. Clicking on “x” will toggle it to a “d” indicating that the value is a decimal number.

USART Master Synchronous Communications

FIGURE 9-7: USART SYNCHRONOUS MASTER SCRIPT BUILDER – SCRIPT DETAIL



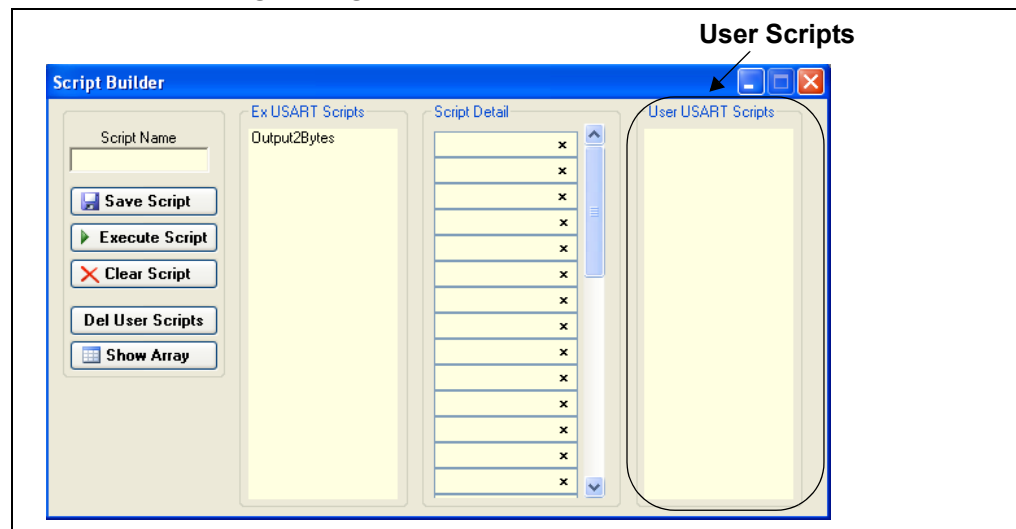
9.7.4 User Scripts

The fourth column contains User Scripts Detail as shown in Figure 9-8. User scripts that are created, named, and saved are displayed in the User Scripts column.

To load a user script from the User Scripts column into the Script Detail column, the user can double click or right click and select from the local menu.

User Scripts can be deleted by right clicking and selecting Delete Script from the local menu.

FIGURE 9-8: USART SYNCHRONOUS MASTER SCRIPT BUILDER – USER SCRIPTS



9.7.5 Creating a Script

Scripts are created by placing the cursor into the Script Detail column and right clicking. A local menu will be displayed as shown in Figure 9-9. Select from the choice of commands or script macro commands.

PICKit™ Serial Analyzer User's Guide

The sequence of macro commands are executed from top to bottom. Macro commands are entered by right clicking in the box and selecting from the local menu as shown in Figure 9-9.

Macro commands are entered according to the sequence of events as defined by the USART Synchronous Master protocol. Studying the example scripts is a good way to learn the sequence of events. The example scripts can also be modified and saved under a different name.

CAUTION

The choice of macro commands is very flexible. Therefore, the correctness of the script has to be verified by the user. The PICKit Serial Analyzer program does not verify the correctness of the script.

A complete listing of the available macro commands is given in Table 9-2. The macro command abbreviation will be displayed in the Transactions Window. The Transactions window keeps a running log of the commands and data sent to and from the target device.

FIGURE 9-9: USART SYNCHRONOUS MASTER SCRIPT BUILDER – CREATING A SCRIPT

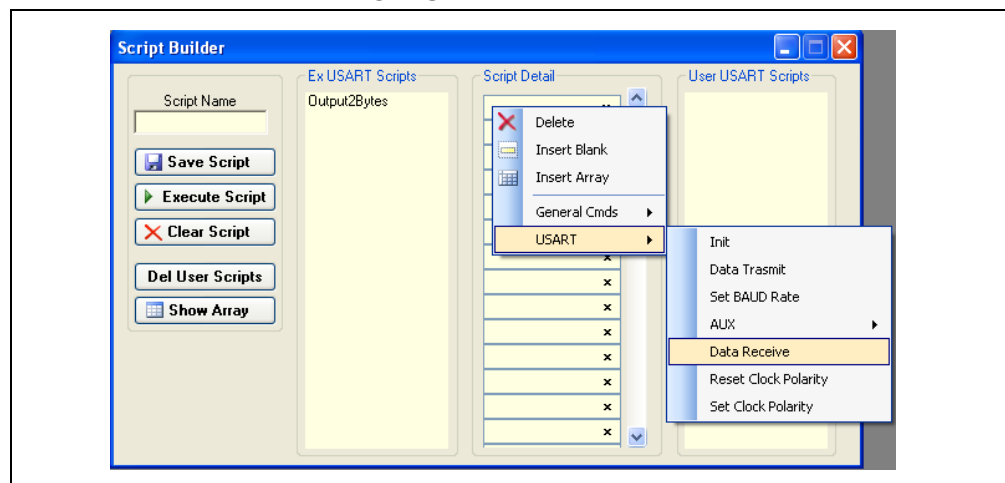


TABLE 9-2: USART SCRIPT MACRO COMMANDS

Macro Command	Command Abbreviation	Description
USDATATX	[TX]	Transmit data. Next byte is the byte count, followed by the data.
USDATARX	[RX]	Receive data. Next byte is the byte count.
USDATARXEN	[ER]	Enable Receive data monitor
USDATARXDS	[DR]	Disable Receive data monitor
USBREAKTX	[BK]	Send Break
USBRKDATTX	[BKD]	Send Break, then data byte. Next byte is the data byte.
USBAUD	[BD]	Set BAUD Rate. Next byte is BAUD (LSB) followed by BAUD (MSB).
US9BITSET	[9S]	Set 9-bit Data mode
US9BITRST	[9R]	Reset 9-bit Data mode (sets 8-bit)
USCLKPOLSET	[CS]	Set CLOCK POLARITY bit
USCLKPOLRST	[CR]	Reset CLOCK POLARITY bit

USART Master Synchronous Communications

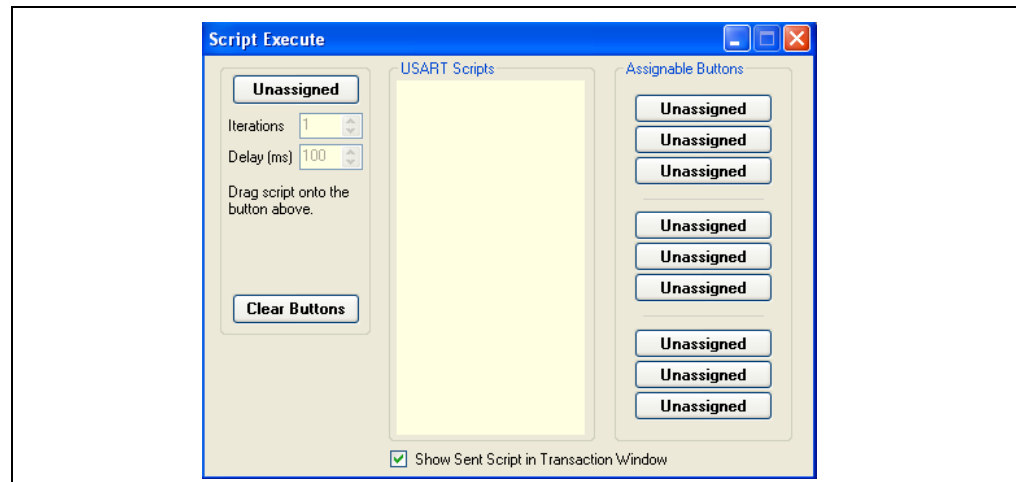
TABLE 9-2: USART SCRIPT MACRO COMMANDS (CONTINUED)

USINIT	[I_]	Initialize USART controller
USRESET	[RE]	Reset USART controller.
USAUX1RST	[A1RST]	Reset Aux1
USAUX1SET	[A1RST]	Set Aux1
USAUX1OUT	[A1OUT]	Set Aux1 direction to Output
USAUX1IN	[A1IN]	Set Aux1 direction to Input
USAUX1W0	[A1W0]	Aux1 Wait 0
USAUX1W1	[A1W1]	Aux1 Wait 1
USAUX2RST	[A2RST]	Reset Aux2
USAUX2SET	[A2RST]	Set Aux2
USAUX2OUT	[A2OUT]	Set Aux2 direction to Output
USAUX2IN	[A2IN]	Set Aux2 direction to Input
USAUX2W0	[A2W0]	Aux2 Wait 0
USAUX2W1	[A2W1]	Aux2 Wait 1

9.8 SCRIPT EXECUTE

The Script Execute window is shown in Figure 9-10. Once scripts are created using the Script Builder, they can be assigned to buttons in the Script Execute window. This makes a convenient window to execute multiple scripts either individually or iteratively. Script executing will be logged in the Transactions window.

FIGURE 9-10: USART ASYNCHRONOUS SCRIPT EXECUTE



9.8.1 Assignable Buttons

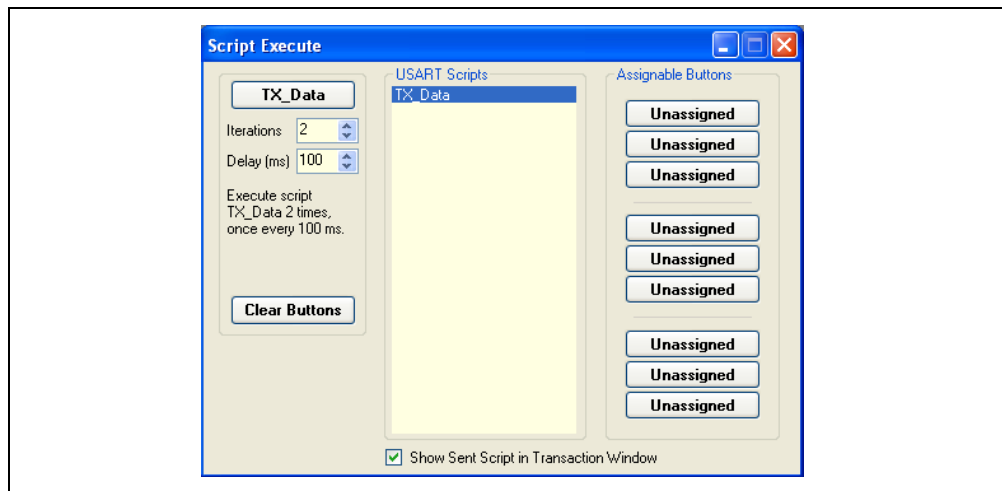
User created scripts will be displayed in the central USART Scripts column. To assign a script to a button, click on the script name and drag it to the desired Assignable Buttons in the right column. The script will be executed once each time the button is clicked.

The Assignable Buttons can be cleared by clicking on the **Clear Buttons** button.

9.8.2 Iteration

Scripts can be executed a user defined number of times at a specified interval of time. Figure 9-11 shows an example. A script named TX_Data has been assigned to the **Iteration** button in the left column. The number of iterations is entered in the Iterations box and the delay in millisecond in the Delay box. A summary of the iterations is displayed in the left column. The macro is executed when the **Iteration** button is clicked.

FIGURE 9-11: USART ASYNCHRONOUS SCRIPT EXECUTE – EXAMPLE



Chapter 10. User Defined Templates

10.1 INTRODUCTION

User Defined Templates extend User Scripts by interpreting the data read from the target device and displaying it in a human readable form. The conversion formula is:

EQUATION 10-1: CONVERSION FORMULA

$$\text{Read Value} * \text{Slope} + \text{Offset} = \text{Display Value}$$

For example, an 8-bit ADC value is read, and we desire a displayed value in voltage, 0 to 5 Volts. The 8-bit ADC value (read value) can be 0 to 256 decimal. The ADC reference voltage is 5 VDC. The slope is the constant value used to convert between the ADC read value and the desired display value. In this example:

EQUATION 10-2: SLOPE EXAMPLE

$$\text{Slope} = \text{ADC Reference Voltage} / \text{8-bit ADC} = 5 / 256 = 0.01953125$$

Using a User Defined Template, the interpretation of the ADC value can be displayed in volts DC.

10.2 HIGHLIGHTS

This chapter discusses:

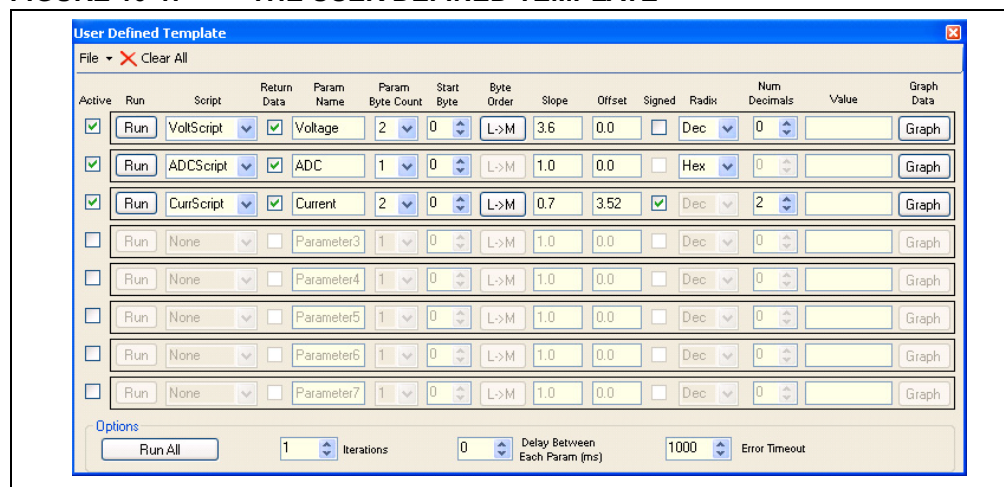
- Creating and Using Templates
- My Templates
- Graphing Data

10.3 CREATING AND USING TEMPLATES

Create a User Define Template by selecting *User Define Templates > Display Template* from the menu bar. The User Defined Template window is displayed as shown in Figure 10-1, showing three active scripts.

Note: First, a user define script(s) must be created using the Script Builder window for the selected serial communications mode. Refer to the respective serial communications chapter under the Script Builder section.

FIGURE 10-1: THE USER DEFINED TEMPLATE



Create the Parameter Template by filling in each line. Each line is a single parameter, a value read converted to value displayed. The *Inputs* below define how a particular parameter is interpreted.

- **Active** – Allows you to activate or deactivate a parameter. This must be active before you can fill in the rest of the data. The purpose of this checkbox is to allow run time activation or deactivation of a script.
- **Run** – Press this button to run your script and display data after you have completed filling out the parameter template line.
- **Script** – Select a user defined script from the drop down box. This script must first be created using the Script Builder for the respective communications mode.
- **Return Data** – Indicate if data is received (Yes) or not (No)
- **Parameter Name** – Enter a parameter name. This is for reference only and does not affect the data retrieved or displayed.
- **Parameter Byte Count** – The number of bytes read (1, 2, or 4)
- **Start Byte** – Select which byte (of a possible series of bytes) that will be used for the display
- **Byte Order** – Select Least Significant to Most Significant or Most Significant to Least Significant. If the parameter is only one byte long, this option will not be available.
- **Slope** – Conversion value (read value * slope + offset = displayed value)
- **Offset** – Offset of read value (read value * slope + offset = displayed value)
- **Signed/Unsigned** – Select between Signed or Unsigned read value (this option is only available if the format is set to Dec)
- **Format** – Binary, Decimal, Hexadecimal, or BCD (Binary Coded Decimal)
- **# of Dec. pts** – Select number of decimal points in the displayed value (this option is only available if the format is set to Dec)

User Defined Templates

- Graph – allows the data to be graphed during run time. The popup graph may be resized as needed.

Once the parameter(s) are entered, they can be saved as a User Defined Parameter file (*.udp).

- File->Open – Open an existing User Defined Parameter file (*.udp)
- File->Save – Save or replace a User Defined Parameter file (*.udp)
- Clear All – Resets (clears) all parameter boxes

OPTIONS

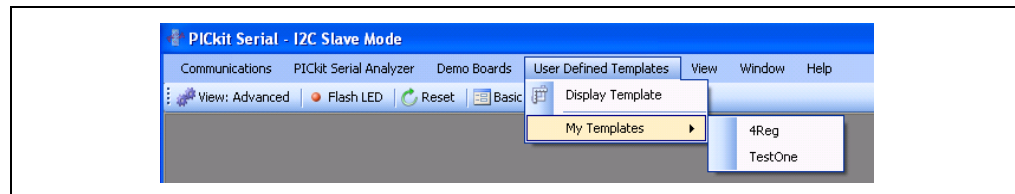
- Run All – Runs all parameter scripts above that are marked as active.
- Iterations – How many times you wish to run the set of active parameters.
- Delay – time in between each parameter.
- Error Timeout – time the GUI will wait for a completed response to the script. If you are running long scripts – you may wish to increase this value.

10.4 MY TEMPLATES

When the User Defined Parameter file has been saved, it will be displayed under the User Defined Templates > My Templates on the menu bar, as shown in Figure 10-2.

Note: The User Defined Parameter file (*.udp) must be stored in the same directory as the PICkit Serial Analyzer executable (by default C:\Program Files\Microchip\PICkit Serial Analyzer). Otherwise it will not be displayed under the My Templates menu selection.

FIGURE 10-2: SELECTING MY TEMPLATES

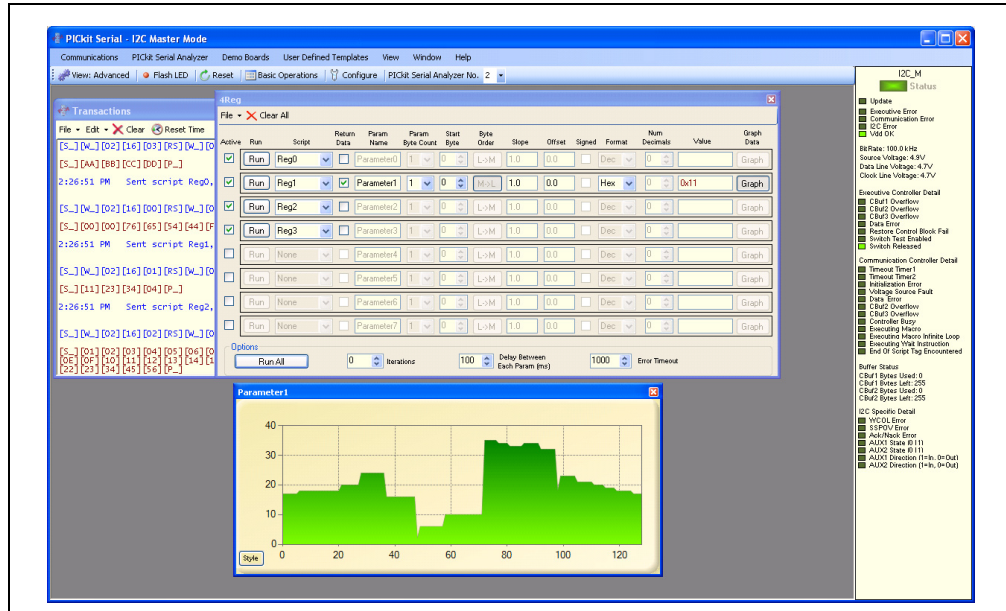


This implementation of the User Defined Template differs from the one in the prior release of PICkit Serial Analyzer in that you create and run the template on the same form. This allows for a great deal of run time flexibility. For example, you can change any of the *Inputs* from section 10.3 while you are running the scripts in a loop.

10.5 GRAPHING DATA

You may also graph any of the data being displayed in real time; simply press the Graph button for the corresponding parameter. The graph may be resized as desired. See Figure 10-3. The style button on the bottom of the graph toggles the view between an area, line, column, or point plot.

FIGURE 10-3: GRAPHING DATA



Chapter 11. PICKit™ Serial Analyzer Firmware

11.1 INTRODUCTION

This chapter explains the internal operations of the PICKit™ Serial Analyzer firmware. The source code is available on the PICKit Serial Analyzer CD-ROM at D:\PICKit Serial Analyzer\Firmware.

11.2 HIGHLIGHTS

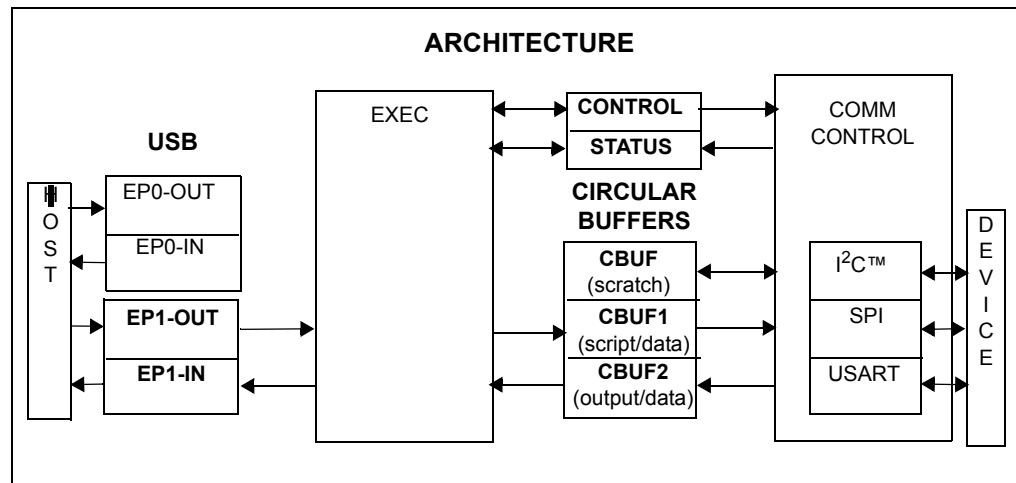
This chapter discusses:

- EXEC
- COMM
- I²C Communications
- I²CS Communications
- SPI/MICROWIRE Communications
- USART Communications
- LIN Communications

11.3 OVERVIEW

Internally, the PICKit™ Serial Analyzer operates two firmware state controllers running in parallel – EXEC (executive) and COMM (communication). The EXEC controller is responsible for overall PICKit™ Serial Analyzer configuration, moving data to/from the host (via USB) and moving data to/from the COMM controller (via RAM buffers). The COMM controller supervises serial communication with the target device. This includes both configuring the necessary communication hardware and executing a sequential 'script' defining a serial 'transaction'.

FIGURE 11-1: ARCHITECTURE



PICKit™ Serial Analyzer User's Guide

Data streams associated with PICKit™ Serial Analyzer are formatted with markers called "TAG" bytes. A TAG may be stand-alone or accompanied by data. As seen in Table 11-1, the data stream for each state controller has a set of associated TAG bytes defined in detail later in the document.

EXEC manages the interface with the host. The data stream sent to PICKit™ Serial Analyzer from the host is encoded with ECMD TAGs and the data stream returned to the host from PICKit™ Serial Analyzer uses EDATA TAGs. Likewise, the COMM controller utilizes another set of predefined TAG(s) - CCMD TAGs (outgoing scripts) and CDATA TAGs (returning data). EXEC has no knowledge of COMM TAG(s) but simply transports data blindly between the host and COMM using EXEC TAG(s). RAM buffers are used as conduits to exchange data with COMM. Data destined for COMM is queued in RAM buffer 1 (CBUF1). Data returning from COMM is funneled through RAM buffer 2 (CBUF2).

The PICKit™ Serial Analyzer is designed to facilitate continuous 'spooling' of data to/from the external serial device. Separate 255-byte circular buffers are maintained for both outgoing data (scripts/data) and returning data. The USB interface is not permitted to be a bottleneck in the PICKit™ Serial Analyzer operation. EXEC processes each USB packet immediately. If data is destined for a RAM buffer, the host is responsible to insure adequate room is available in the buffer before the data is sent to avoid a fatal 'overrun' error. Returning data is queued in the appropriate circular buffer until retrieved by EXEC, tagged and sent to the host. It is possible to overrun the return buffer under some circumstances but should be rare. EXEC can interleave EXEC data with COMM data as necessary.

The PICKit™ Serial Analyzer maintains fixed-length blocks of data for CONTROL and STATUS. The CONTROL_BLOCK provides 'static' configuration information. The STATUS_BLOCK is a snapshot of the PICKit™ Serial Analyzer operation. Each block is divided into three sections corresponding to EXEC, COMM (common to all protocols) and COMM (specific to the active serial protocol).

TABLE 11-1: TAG BYTE TYPES

TAG types	Definition
ECMD	EXEC command TAG(s) – interpreted by EXEC
EDATA	EXEC data TAG(s) – generated by EXEC
CCMD	COMM command TAG(s) – interpreted by COMM
CDATA	COMM data TAG(s) – generated by COMM

The other major blocks are the control memory block and the status memory block. The control block is used to configure the PICKit™ Serial Analyzer. The control block is divided into the following three sections:

- EXEC module configuration
- Generic COMM module configuration
- Protocol specific communication configuration

This third section will change depending on the protocol. The Status block keeps the status of various flags and is similarly divided into three sections.

TABLE 11-2: CONTROL BLOCK

TAG Bytes	Definition
0-7	EXEC section
8-15	Generic COMM section
16-23	Protocol specific communication section

TABLE 11-3: STATUS BLOCK

TAG Bytes	Definition
0-3	EXEC section
4-11	Generic COMM section
12-19	Protocol specific communication section

11.4 EXEC

The EXEC module will directly decode the data stream from the host. The list of different commands is shown in Table 11-4. Every data stream from the software to the PICKit™ Serial Analyzer begins with one of the following EXEC command TAG bytes.

TABLE 11-4: EXEC COMMAND (ECMD) TAG BYTES

TAG/ECMD	LEN	Name	Description
0x00	1	END OF DATA	Marks the end of valid data
			0 TAG
0x01	1	COMMAND	Executive command
			0 TAG
			1 Command byte (see Table 11-5)
0x02	25	CONTROL_BLOCK_WRITE	Write control block to PIC® MCU
			0 TAG
			1:24 Control block
0x03	N+2	CBUF1_WRITE	Write data to circular buffer 1
			0 TAG
			1 Byte count (N)
			2 : N+1 Data
0x04	N+2	CBUF2_WRITE	Write data to circular buffer 2
			0 TAG
			1 Byte count (N)
			2 : N+1 Data
0x05	N+2	CBUF3_WRITE	Write data to circular buffer 3
			0 TAG
			1 Byte count (N)
			2 : N+1 Data
0x06	2	LED1_CONFIG	Configure LED 1
			0 TAG
			1 Configuration Byte
0x07	2	LED2_CONFIG	Configure LED 2
			0 TAG
			1 Configuration byte

TAG byte 0x01 signifies that the following byte is one of the commands listed in Table 11-5.

PICkit™ Serial Analyzer User's Guide

TABLE 11-5: TAG BYTE 0x01 COMMAND CODES

CMD	Description
0x00	Master Reset: EXEC Reset, COMM Reset (idled)
0x01	COMM initialization: COMM is initialized as defined by CONTROL_BLOCK
0x02	Request EXEC_STATUS_PACKET (Ref. Table 11-6)
0x03	Save CONTROL_BLOCK to EEPROM
0x04	Restore CONTROL_BLOCK from EEPROM
0x05	Flush CBUF2
0x06	COMM Reset: rest buffers, clear status block (COMM hardware is not re-initialized)

TABLE 11-6: EXEC STATUS PACKET

Byte	LEN	Type	Description
0	1	0x88	PACKET ID = 0x01
2	4	0x81	FIRMWARE VERSION
5	29	0x82	CONTROL BLOCK
30	50	0x83	STATUS BLOCK
51	57	0x84	CBUF STATUS

TAG byte 0x02 writes the 24-byte CONTROL_BLOCK (the EXEC portion of the control block appears in Table 11-7).

TABLE 11-7: EXEC CONTROL BLOCK

Byte	Bit	Description
0	7:0	EXEC control bits
	0	
	1	
	2	
	3	
	4	1 = Disable default behavior – LED2
	5	1 = Disable default behavior – LED1
	6	1 = Flush CBUF2 on count [e.g. CBUF2 >= N bytes then flush]
7	1 = Flush CBUF2 on time intervals	
1	7:0	
	0	1 = Enable switch test
	1	n/a
	2	n/a
	3	n/a
	4	n/a
	5	n/a
	6	n/a
7	n/a	
2	7:0	
3	7:0	CBUF2 flush count threshold [e.g., CBUF2 > N bytes then flush]
4	7:0	CBUF2 flush interval [res: 409 μs, min: 409 μs, max: 104 msec] a value of '0' defaults to '1', (i.e., the minimum)
5	7:0	
6	7:0	
7	7:0	

TAG bytes 0x03, 0x04 and 0x05 write data bytes to their respective script buffers. In the current architecture, Script Buffer 1 (CBUF1) is used to store communication commands that will be fetched and executed by the COMM block. So, TAG byte 0x03 is used to delineate data that is to be sent to the script buffer including communication protocols to be sent to the unit under test. TAG bytes 0x04 and 0x05 are typically not used. TAG bytes 0x06 and 0x07 configure the LEDs as follows:

TABLE 11-8: LED CONFIGURATION

Description	Mode M = CFG[7:6]	State S = CFG[5]	Time T = CFG[4:0]
Set immediate	00	1 = On, 0 = Off	N/A
Blink once – On or Off	10	1 = On, 0 = Off	T + 1 units
Blink continuous	11	initial state: 1 = On, 0 = Off	T + 1 units
Reserved	01		

Legend: Time "unit" = 50 ms

Note: Set LED to blink continuously, on/off time = 100 ms.
CFG = 0xC1 (M = b'11', S = 0, T = 1)

EXEC data TAG bytes identify data streams sent from the EXEC block back to the host software.

TABLE 11-9: EXEC TAG (EDATA) BYTES

TAG/EDATA	LEN	Name	Description
0x80	1	END OF DATA	Marks the end of valid data
			0 TAG
0x81	1	FIRMWARE_VERSION	Firmware version
			0 TAG
			1 Data: 20-byte STATUS_BLOCK
			2 Date: major version
0x82	25	CONTROL_BLOCK_DATA	Control block contents
			0 TAG
			1:24 Data: CONTROL_BLOCK
0x83	21	STATUS_BLOCK_DATA	Status block contents
			0 TAG
			1:20 Data: minor version
0x84	7	CBUF_STATUS	Status block contents
			0 TAG
			1 Data: CBUF1: # bytes used
			2 Data: CBUF1: # bytes unused
			3 Data: CBUF2: # bytes used
			4 Data: CBUF2: # bytes unused
			5 Data: CBUF3: # bytes used
			6 Data: CBUF3: # bytes unused
0x85	N+2	CBUF1_DATA	Data from CBUF1
			0 TAG
			1 Byte count
			2 : N+1 Data: from CBUF1
0x86	N+2	CBUF2_DATA	Data from CBUF2
			0 TAG
			1 Byte count
			2 : N+1 Data: from CBUF2

PICkit™ Serial Analyzer User's Guide

TABLE 11-9: EXEC TAG (EDATA) BYTES (CONTINUED)

TAG/EDATA	LEN	Name	Description	
0x87	N+2	CBUF3_DATA	Data from CBUF3	
			0	TAG
			1	Configuration byte
			2 : N+1	Data: from CBUF3
0x88	2	PACKET_ID	Packet number	
			0	TAG
			1	Data: arbitrary packet ID#

TAG byte 0x80 means the transaction is over. TAG byte 0x81 signifies that the following data is the firmware version. 0x82 signifies that the data following is the control block (CONTROL_BLOCK). 0x83 signifies that the data following data is the status block (STATUS_BLOCK). The EXEC portion of STATUS_BLOCK is shown in Table 11-10.

TABLE 11-10: EXEC STATUS BLOCK

Byte	Bit	Description
EXEC		
0	7:0	EXEC status
	0	
	1	CBUF1 overflow
	2	CBUF2 overflow
	3	CBUF3 overflow
	4	Data error (e.g. illegal TAG, missing TAG-dependent data, etc.)
	5	Restore control block failed – defaults used
	6	
1	7:0	EXEC status flags
	0	SWITCH test active
	1	SWITCH state i.e. 0: depressed, 1: released
	2	
	3	
	4	
	5	
	6	
2	7:0	
	3	7:0

TAG byte 0x84 is the script buffer status which shows availability of memory in each buffer. TAG bytes 0x85, 0x86 and 0x87 identify data streams coming from RAM buffers CBUF1, CBUF2 and CBUF3, respectively. RAM buffer 2 (CBUF2) is used in this architecture to queue the data stream from the COMM module to be sent to the host software, thus TAG byte 0x86 identifies this stream.

TAG bytes 0x85 and 0x87 are typically unused.

11.5 COMM

The COMM module will decode TAG bytes from the data stream in the script buffer. The TAG bytes may represent commands to internally configure the PICkit™ Serial Analyzer and report status, or the TAG bytes may be protocol specific identification that

PICKit™ Serial Analyzer Firmware

needs to be translated into the device under test's protocol and communicated to the device. The COMM module performs both of these functions. This section will describe the COMM TAGs common to all supported serial protocols. Table 11-11 describes the TAGs (CCMD) used in the data stream to the COMM controller. Table 11-12 describes TAGs (CDATA) used in the data stream from the COMM controller

TABLE 11-11: COMM SCRIPT COMMAND TAG BYTES

TAG/CCMD	LEN	Name	Description
0x00 – 0x0F	16	RESERVED	Reserved
0x10	3	Wait-1	Wait for time interval
			0 TAG
			1 Time (LSB)
			2 Time (MSB) [res: 409.6 μ sec, max: 26.843 sec]
0x12	2	LED1	Configure LED1
			0 TAG
			1 LED Configuration byte
0x13	2	LED2	Configure LED2
			0 TAG
			1 LED Configuration byte
0x15	3	TIMEOUT_AB0_SET	Set time-out AB0
			0 TAG
			1 Time-out value (LSB)
			2 Time-out value (MSB) [res: 409.5 ms, max: 26.843 sec]
0x16	1	TIMEOUT_AB0_KILL	Kill/disable time-out AB0
			0 TAG
0x17	3	TIMEOUT_AB1_SET	Set time-out AB1
			0 TAG
			1 Time-out value (LSB)
			2 Time-out value (MSB) [res: 409.6 ms, max: 26.843 sec]
0x18	1	TIMEOUT_AB1_KILL	Kill/disable timeout AB1
			0 TAG
0x19	1	MACRO_CLEAR	Start macro definition
			0 TAG
0x1A	N+2	MACRO_DATA_ADD	Add bytes to macro
			0 TAG
			1 Byte count (N)
			2 Data
			N+1 Data
0x1B	3	MACRO_RUN	Run macro
			0 TAG
			1 Loop count [0 = indefinitely]
			2 Loop count [0 = indefinitely]
0x1C	1	END_OF_SCRIPT	Mark end of script
			0 TAG
0x1D	1	MACRO_DATA_START	Mark start of "macro" data block
			0 TAG
0x1E	1	MACRO_DATA_END	Mark end of "macro" data block
			0 TAG

PICkit™ Serial Analyzer User's Guide

TABLE 11-11: COMM SCRIPT COMMAND TAG BYTES (CONTINUED)

TAG/CCMD	LEN	Name	Description	
0x1F	2	MARKER	Script "marker"	
			0	TAG
			1	Marker
0x20	1	EVENT_TIMER_RESET	Event timer Reset	
			0	TAG

TABLE 11-12: COMM SCRIPT DATA TAG BYTES

TAG/CDATA	LEN	Name	Description	
0x00 – 0x0F	16	RESERVED	RESERVED	
0x10	2	DATA_BYTE	Data byte follows	
			0	TAG
			1	data
0x11	N+2	DATA_BYTES	Data bytes follow	
			0	TAG
			1	Byte count (N)
			2	Data
			N+1	Data
0x12	2	EVENT_MACRO_LOOP	Macro loop count milestone message	
			0	TAG
			1	Loop number
0x13	3	EVENT_TIME	Time marker for previous event	
			0	TAG
			1	Time LSB
			2	Time MSB [res: 409 usec, max: 26.8 sec]
0x14	1	EVENT_TIME_ROLLOVER	Event timer rollover	
			0	TAG
0x15	3	EVENT_MACRO_DONE	Macro loop complete	
			0	TAG
			1	Loop count (LSB)
			2	Loop count (MSB)
0x16	1	EVENT_MACRO_ROLLOVER	Macro loop count rollover (65536) (useful with infinite loop count)	
			0	TAG
0x17	1	EVENT_MACRO_ABORT	Macro execution was aborted by bit in command block	
			0	TAG
0x18	1	EVENT_TIMEOUT_AB0	Timer AB0 timeout	
			0	TAG
0x19	1	EVENT_TIMEOUT_AB1	Timer AB1 timeout	
			0	TAG
0x1A	2	EVENT_STATUS_ERR	Status error	
			0	TAG
			1	STATUS_BLOCK[04]
0x1B	1	EVENT_END_OF_SCRIPT	"End-of-Script" TAG encountered	
			0	TAG
0x1C	2	MARKER	"Marker" tag encountered	
			0	TAG
			1	MARKER

TABLE 11-13: COMM SCRIPT CONTROL BLOCK

Byte	Bit	Description
COMM: GENERAL		
8	7:0	COMM mode: 00: IDLE 01: I ² CM 02: SPI-M 04: USART-A 05: USART-SM 07: I ² CS 0A: LIN 0B: MWIRE-M
9	7:0	COMM control bits
	0	1 = Enable event markers – global
	1	1 = Enable event markers – time stamp
	2	n/a
	3	n/a
	4	1 = Enable PULLUPS
	5	1 = VSRC: On
	6	1 = VSRC: variable
7	1 = Abort macro execution	
10	7:0	Bit flags
	0	1 = Event marker enable: macro loop
	1	1 = Event marker enable: macro loop 65536
	2	1 = Event marker enable: macro loop done
	3	1 = Event marker enable: time-out AB0
	4	1 = Event marker enable: time-out AB1
	5	1 = Event marker enable: status error (STATUS_BLOCK[04] != 0)
	6	1 = (Unassigned)
7	1 = (Unassigned)	
11	7:0	N/A
12	7:0	VSRC 8-bit setting
13	7:0	VSRC Fault Threshold
14	7:0	N/A
15	7:0	N/A

TABLE 11-14: COMM SCRIPT STATUS BLOCK

Byte	Bit	Description
COMM: GENERAL		
4	7:0	COMM status: error
	0	Time-out AB0
	1	Time-out AB1
	2	COMM initialization error
	3	VSRC fault
	4	Data error (e.g., unrecognized TAG, missing data, etc.)
	5	Output buffer overrun (CBUF2)
	6	Output buffer overrun (CBUF3)
5	7:0	COMM status: informational
	0	Controller busy
	1	Executing MACRO
	2	Executing MACRO – infinite loop
	3	Executing WAIT instruction
	4	
	5	
	6	
7	“End-of-Script” TAG encountered	
6	7:0	COMM mode
7	7:0	VSRC Measurement (0-255)
8	7:0	
9	7:0	
10	7:0	
11	7:0	

11.6 I²C COMMUNICATIONS

I²C refers to the Master mode of the I²C protocol. See the I²C specification for protocol details. In this mode, the PICkit™ Serial Analyzer will master the I²C bus only. It will not respond as a slave. The control block of memory dedicated to communication allows for setting of the bit rate and displaying event markers in the software windows.

TABLE 11-15: CONNECTOR PINOUT IN I²C MODE

Pin	Description
1	—
2	+V
3	GND
4	SDA
5	SCL
6	—

TABLE 11-16: I²CM CONTROL BLOCK

Byte	Bit	Description
COMM: I²CM		
16	7:0	Bit flags
	0	1 = event marker enable: Start bit
	1	1 = event marker enable: Stop bit
	2	1 = event marker enable: Restart bit
	3	1 = event marker enable: ack/nack tx
	4	n/a
	5	1 = event marker enable: ack/nack rx
	6	n/a
17	7:0	Bit flags
	0	1 = event marker enable: read byte
	1	1 = event marker enable: transaction error
	2	1 = event marker enable: status error (STATUS_BLOCK[12] != 0)
	3	
	4	
	5	
	6	
18	7:0	
19	7:0	
20	7:0	
21	7:0	
	0	AUX1 – default state (0 1)
	1	AUX2 – default state (0 1)
	2	AUX1 – direction (1=IN, 0=OUT)
	3	AUX2 – direction (1=IN, 0=OUT)
	4	n/a
	5	n/a
	6	n/a
	7	n/a
22	7:0	
23	7:0	BIT RATE = (FOSC / (4 * (X + 1))) min: 0, max: 127 49 => 100k bps 127 => 39k bps

PICKit™ Serial Analyzer User's Guide

TABLE 11-17: I²CM STATUS BLOCK

Byte	Bit	Description
COMM: I²CM		
12	7:0	Bit flags: error status
	0	WCOL error
	1	SSPOV error
	2	received NACK when ACK was expected
	3	
	4	
	5	
	6	
13	7:0	Bit flags: info status
	0	
	1	
	2	
	3	
	4	
	5	
	6	
14	7:0	
	0	
	1	
	2	
	3	
	4	
	5	
	6	
15	7:0	
	0	
	1	
	2	
	3	
	4	
	5	
	6	
16	7:0	
	0	AUX1 – state (0 1)
	1	AUX2 – state (0 1)
	2	AUX1 – direction (1=IN, 0=OUT)
	3	AUX2 – direction (1=IN, 0=OUT)
	4	n/a
	5	n/a
	6	n/a
17	7:0	
	0	
	1	
	2	
	3	
	4	
	5	
	6	
18	7:0	
	0	
	1	
	2	
	3	
	4	
	5	
	6	
19	7:0	
	0	
	1	
	2	
	3	
	4	
	5	
	6	
19	7:0	BIT RATE CODE currently in use
	0	
	1	
	2	
	3	
	4	
	5	
	6	

The I²CM TAG/CCMND bytes are used by the host (software) to describe an I²C transaction in 'script' form. The script is sent via USB to the script buffer (CBUF1) and interpreted by the COMM controller to execute the transaction on the I²C bus. The return data stream contains I²C read data as well as 'event marker' TAGs that mark the occurrence of each entity of the I²C transaction (e.g., TAG: 0x81 indicates a "Start" bit)

TABLE 11-18: I²CM 'CMD' TAG BYTES

TAG/CCMD	LEN	Name	Description
0x80	1	I ² CM_INIT	Initialize master
			0 TAG
0x81	1	I ² CM_START	Issue I ² C™ Start
			0 TAG
0x82	1	I ² CM_STOP	Issue I ² C™ Stop
			0 TAG
0x83	1	I ² CM_RESTART	Issue I ² C™ Restart
			0 TAG
0x84	N+2	I ² CM_WRITE_BYTES	Write bytes
			0 TAG
			1 byte count (N)
			2 data byte
0x85	2	I ² CM_READ_BYTES	Read bytes – ACK all bytes
			0 TAG
			1 byte count (N)
			N+1 data byte
0x86	1	I ² CM_READ_BLOCK	Read block – ACK all bytes
			0 TAG
0x87	2	I ² CM_BIT_RATE	Set I ² C™ bit rate
			0 TAG
			1 bit rate
0x88	1	I ² CM_RESET	Reset MSSP module
			0 TAG
0x89	2	I ² CM_READ_BYTES_NLB	Read bytes – NACK last byte
			0 TAG
			1 byte count (N)
0x8A	1	I ² CM_READ_BLOCK_NLB	Read block – NACK last byte
			0 TAG
0x90	1	I ² CM_AUX1_RST	AUX1: 0
			0 TAG
0x91	1	I ² CM_AUX1_SET	AUX1: 1
			0 TAG
0x92	1	I ² CM_AUX1_OUT	AUX1 direction: OUTPUT
			0 TAG
0x93	1	I ² CM_AUX1_IN	AUX1 direction: INPUT
			0 TAG
0x94	1	I ² CM_AUX1_WAIT_0	wait AUX1 == 0
			0 TAG

PICkit™ Serial Analyzer User's Guide

TABLE 11-18: I²CM 'CMD' TAG BYTES (CONTINUED)

0x95	1	I ² CM_AUX1_WAIT_1	wait AUX1 == 1
			0 TAG
0x96	1	I ² CM_AUX2_RST	AUX2: 0
			0 TAG
0x97	1	I ² CM_AUX2_SET	AUX2: 1
			0 TAG
0x98	1	I ² CM_AUX2_OUT	AUX2 direction: OUTPUT
			0 TAG
0x99	1	I ² CM_AUX2_IN	AUX2 direction: INPUT
			0 TAG
0x9A	1	I ² CM_AUX2_WAIT_0	wait AUX2 == 0
			0 TAG
0x9B	1	I ² CM_AUX2_WAIT_1	wait AUX2 == 1
			0 TAG

An example of a data stream in the script buffer that would direct the COMM back to communicate in I²C to the unit.

Under test is as follows:

0x81	I ² CM_START
0x84	I ² CM_WRITE_BYTES
0x02	Number of bytes
0xA8	I ² C™ address for writing
0x01	I ² C™ command code
0x83	I ² CM_RESTART
0x84	I ² CM_WRITE_BYTES
0x01	Number of bytes
0xA9	I ² C™ address for reading
0x89	I ² CM_READ_BYTES_NLB
0x01	Number of bytes
0x82	I ² CM_STOP

The script (above) is interpreted as follows. TAG 0x81 instructs the COMM module to generate an I²C 'Start' bit on the I²C bus. TAG 0x84 indicates 2 bytes will be transmitted following the start – 0xA8 and 0x01. The first byte is the I²C slave address (with write/read bit Reset) and a data/command byte of 0x01. The COMM module does not place any significance on the value of the data bytes but merely transmits them 'blindly' – as instructed. The next TAG 0x83 instructs the COMM module to issue a Restart bit on the I²C bus. TAG and data bytes - 0x84, 0x01, 0xA9 – will cause 1 byte (0xA9) to be transmitted. Here again, the COMM module does not interpret the data – the I²C slave will interpret 0xA9 as an address with write/read bit set. TAG 0x89 followed by data byte 0x01 instructs the COMM module to attempt to read 1 byte from the slave then issue a NACK on the bus. Finally, an I²C 'Stop' bit is issued according to tag 0x82. The resulting I²C transaction looks like this on the bus:

[START][A8][01][RESTART][A9][data byte received][STOP]

As the script is executed, a data stream will be developed using TAGs/CDATA (described in Table 11-19) and returned to the host software via CBUF2.

TABLE 11-19: I²CM 'DATA' TAG BYTES

TAG/ CDATA	LEN	Name	Description
0x80	1	I ² CM_EVENT_START_TX	Start bit event
			0 CDATA-TAG
0x81	1	I ² CM_EVENT_STOP_TX	Stop bit event
			0 CDATA-TAG
0x82	1	I ² CM_EVENT_RESTART_TX	Restart bit event
			0 CDATA-TAG
0x83	1	I ² CM_EVENT_ACK_TX	ACK bit event
			0 CDATA-TAG
0x84	1	I ² CM_EVENT_NACK_TX	NACK bit event
			0 CDATA-TAG
0x85	1	I ² CM_EVENT_ACK_RX	ACK bit event
			0 CDATA-TAG
0x86	1	I ² CM_EVENT_NACK_RX	NACK bit event
			0 CDATA-TAG
0x87	2	I ² CM_EVENT_BYTE_TX	BYTE transmit
			0 CDATA-TAG
			1 data
0x88	2	I ² CM_EVENT_BYTE_RX	BYTE transmit
			0 CDATA-TAG
			1 data
0x89	2	I ² CM_EVENT_XACT_ERR	transaction error
			0 CDATA-TAG
			1 error byte
0x8A	2	I ² CM_EVENT_STATUS_ERR	status error
			0 CDATA-TAG
			1 error byte

11.7 I²C COMMUNICATIONS

The I²C communication mode is a flexible I²C slave that can be configured to respond to one or more bus addresses. Three operating modes (described below) provide several levels of sophistication and operability. In all modes, transactions can be reported to the host using optional PKSA event tags. Additionally, mode-dependant data is available as described below.

TABLE 11-20:

MODE	NAME	DESCRIPTION
0	DEFAULT	basic/mechanical mode of operation in which the PKSA blindly accepts any/all 'write' data and provides canned/default 'read' data in response to any enabled I2C address and all device registers/addresses (data defined in CONTROL_BLOCK).
1	INTERACTIVE	This mode allows the host to orchestrate I2C transactions in 'real time'. This necessarily requires the host to provide 'read' data as needed while the PKSA holds the I2C bus clock line (waiting). 'Write' data is reported to the host via transaction event tags.
2	AUTO	In AUTO mode the PKSA operates autonomously as defined by a 'soft' "SLAVE PROFILE" table stored in PKSA RAM (described below). At any time the host can read and/or update the table contents as needed.

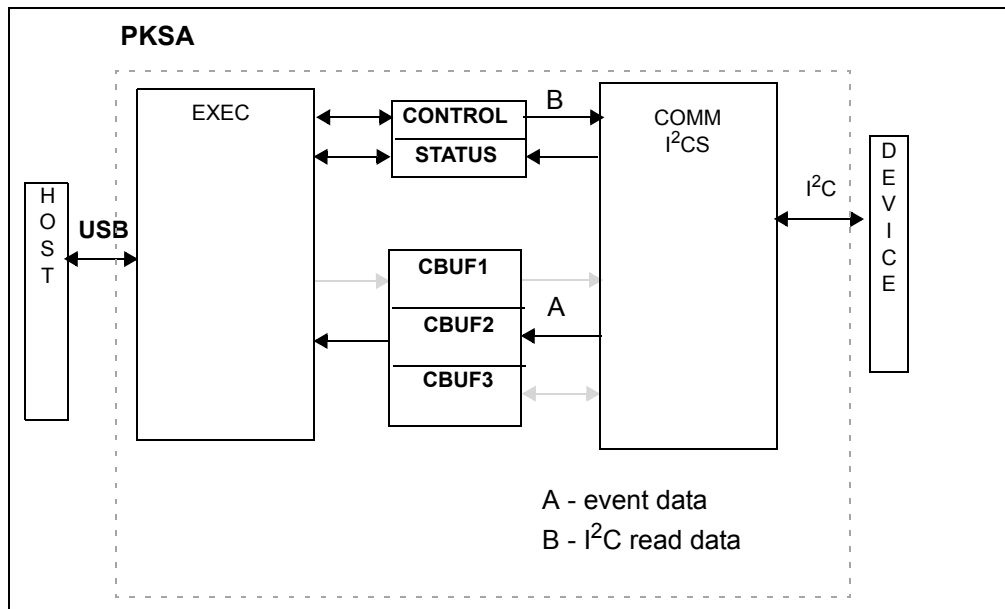
Note: The ability to service multiple slave addresses is hampered by a bug in silicon - versions A3 and before. Slave address MASK should be set to zero ("0").

11.7.1 Modes

11.7.1.1 MODE: DEFAULT

As stated in the introduction, the DEFAULT mode requires little/no configuration and results in minimal transaction latencies. This mode is of limited use in that it does not provide for address-dependant (dynamic) read data. The CONTROL_BLOCK is the source of 'read' data for any/all addresses. Of course optional EVENT markers report I²C transactions to the host.

FIGURE 11-2:



11.7.1.1.1 Operation

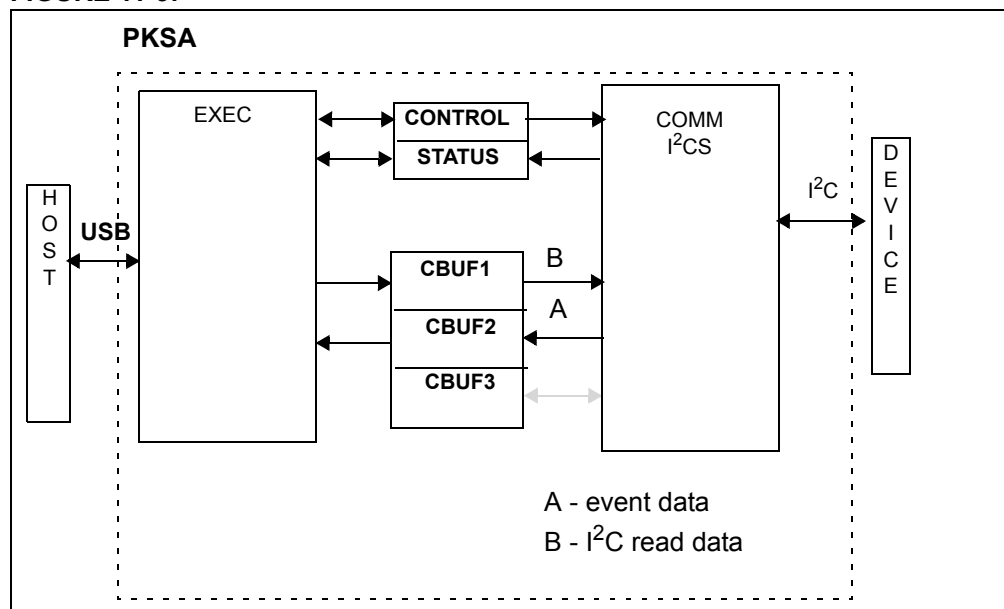
The 'DEFAULT' mode requires very little configuration/initialization.

1. Issue COLD START
2. Write CONFIGURATION BLOCK (with COMM=I²CS and MODE = 0)
3. Issue WARM START

11.7.1.2 MODE: INTERACTIVE

The INTERACTIVE mode provides the greatest flexibility but at the expense of transaction latencies. The host monitors I²C transactions near real time using EVENT markers. Should a transaction require 'read' data (to return to the I²C master), the I²C bus is suspended (by holding the clock line low) until the host supplies data via the standard 'script' buffer.

FIGURE 11-3:



11.7.1.2.1 Operation

Execute the following sequence for initialization:

1. Issue COLD START (Reset)
2. Write CONFIGURATION BLOCK (with COMM=I2CS, MODE=1, EVENT MARKERS enabled)
3. Issue WARM START

The HOST must monitor I²C transactions via the EVENT MARKERS. Should the I²C MASTER request 'read' data, the PKSA will hold the I²C SCK line low and alert the HOST with SDATA TAG: I2CS_DATA_RQ. The HOST must supply data to the PKSA to be returned to the I2CS MASTER using SCMND TAG: I2CS_SEND_BYTES. If more data is supplied than is required for the transaction, the remaining data will be discarded. If an insufficient number of bytes are supplied by the HOST, the PKSA will return all data supplied then hold SCK line low and re-issue SCMND TAG: I2CS_DATA_RQ to the HOST.

11.7.1.3 MODE: AUTO

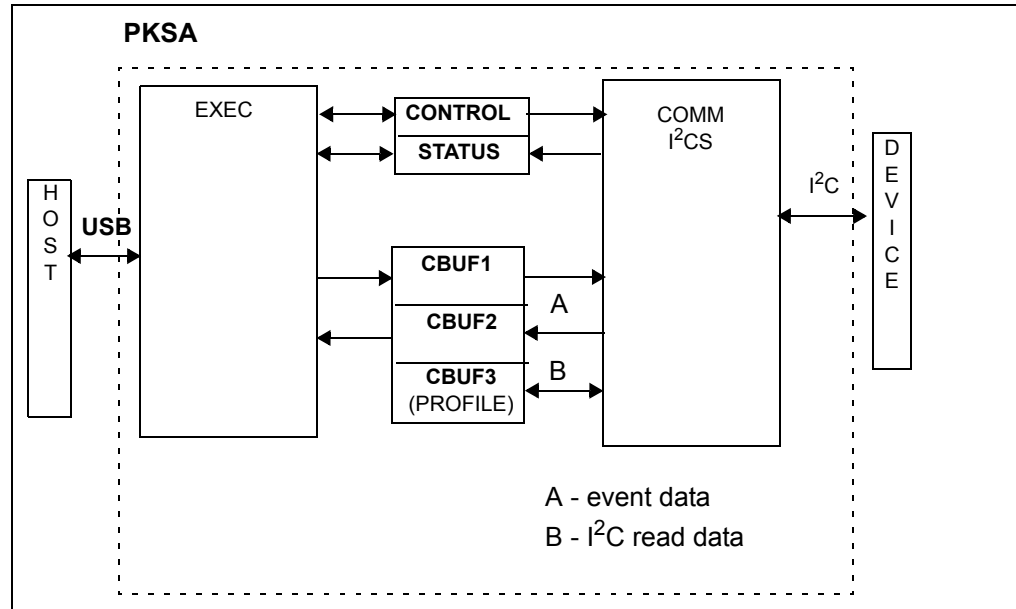
AUTO is the most sophisticated mode allowing the PKSA to operate 'autonomously' from a 'slave profile' loaded in PKSA RAM (CBUF3). The slave profile defines one or more I²C slave addresses and a unique set of device registers for each. The device registers can be defined in two ways – (1) discrete, non-contiguous registers of various lengths or (2) a block of contiguous addresses with flexible access to 1 or more bytes within that block in a single transaction. Additionally, each register can be defined as Read/Write, Read-only or Write-only.

The 2 bytes in the PKSA CONTROL BLOCK, ADDRESS and MASK, defines the set of SLAVE addresses that the PKSA will 'ACKNOWLEDGE'. All addresses excluded by the ADDRESS and MASK will be automatically 'NOT ACKNOWLEDGED'. For all acknowledged address the pre-loaded SLAVE PROFILE dictates the response/behavior.

For I²C transactions involving SLAVE ADDRESSES or REGISTERS not defined in the SLAVE PROFILE, WRITE data is discarded and a READ data is defaulted (0xFF). Similarly, writing to a 'READ-ONLY' register or attempting to write beyond the addressable

limits of the register will cause the write data to be discarded. Reading from a 'WRITE-ONLY' register or attempting to read beyond the addressable limits of the register will result in default data (0xFF).

FIGURE 11-4:



11.7.1.3.1 Operation:

The HOST configures the AUTO mode as follows:

1. Issue COLD START (Reset)
2. Write CONFIGURATION BLOCK (comm=I2CS, mode=2, appropriate EVENT MARKERS enabled)
3. Write SLAVE PROFILE to CBUF3
4. Issue WARM START

The AUTO mode allows the PKSA to operate autonomously (i.e. without HOST intervention). Practically speaking, the HOST GUI will most likely want to reflect activity with the SLAVE registers as it occurs. A couple options can be employed.

Option 1 would call for the HOST to enable I²C transaction EVENT MARKERS detailing each I²C transaction in which the PKSA participated. This requires that the HOST decode/interpret the transactions on a byte-by-byte basis.

Option 2 utilizes the 'register access' EVENT MARKERS (I2CS_EVENT_REG_READ & I2CS_EVENT_REG_WRITE) fired to the HOST when a register within the SLAVE PROFILE is accessed. The fact that a slave register was read is an 'event' but does not represent a change in the data loaded in the SLAVE PROFILE. In the event that a register within the SLAVE PROFILE was written, it may be important for the HOST monitoring the SLAVE PROFILE to fetch the 'new' data from that register for processing/display. Data is retrieved from a register using SCMND TAG: I2CS_REGDATA_READ. The register contents are returned with SDATA TAG:

I2CS_REGDATA.

The register contents can be written or read by the HOST. To insure data integrity of multi-byte registers, the contents are accessed only 'between' I²C transactions. This may cause a slight delay in executing the request or in servicing the next/pending I²C transactions.

11.7.1.3.2 Slave Profile

In AUTO mode the PKSA personality is defined by a “slave profile” downloaded by the host to PKSA RAM i.e. CBUF3. The profile defines one or more I²C slave addresses and a set of one or more device REGISTERS for each. The profile must begin with a SLAVE ADDRESS TAG and be terminated by an ‘END’ TAG.

TABLE 11-21:

TABLE-TAG ⁽¹⁾	DESCRIPTION
0x10	SLAVE ADDRESS
0x2X	REGISTER DEFINITION
0x40	END OF SLAVE PROFILE

Note 1: The upper nibble defines the TABLE-TAG, the lower nibble is reserved for option flags

TABLE 11-22: TABLE-TAG: SLAVE ADDRESS (0X20)

TABLE-TAG: SLAVE ADDR: 0x20
DATA: SLAVE ADDRESS
DATA: (RESERVED)
DATA: (RESERVED)

The SLAVE definition is 4 bytes long (including the TAG). Two bytes are reserved for firmware operation.

TABLE 11-23: TABLE-TAG: REGISTER DEFINITION

TABLE-TAG: REG DEF: 0x4X
DATA: REGISTER ID / # / ADDRESS
DATA: BYTE COUNT = N
DATA: DATA[0]
...
DATA: DATA[N-3]

The length of a REGISTER definition is flexible.

Lower nibble of TABLE-TAG is defined as option bits (refer to table below):

TABLE 11-24:

TABLE-TAG[3]	‘memory block’ of sequential locations beginning with register #
TABLE-TAG[2]	
TABLE-TAG[1]	disable READ
TABLE-TAG[0]	disable WRITE

A ‘memory block’ is a set of N contiguous addresses beginning with ‘REGISTER ID’. An I²C transaction is permitted to access one or more bytes anywhere within the block.

TABLE 11-25: TABLE-TAG: END

TABLE-TAG: END: 0x80

EXAMPLE

The following SLAVE PROFILE defines 2 slave addresses – 2 registers for 1 and 1 register block for the other.

TABLE 11-26:

TABLE TAG SLAVE ADDR	0x20
DATA: SLAVE ADDR	0x16
DATA: (RESERVED)	0x00
DATA: (RESERVED)	0x00
DATA: TABLE-TAG: REGISTER DEF	0x41
DATA: REGISTER #	0x21
DATA: BYTE COUNT	0x02
DATA:	0x34
DATA:	0x12
DATA: TABLE-TAG: REGISTER DEF	0x40
DATA: REGISTER #	0x22
DATA: BYTE COUNT	0x01
DATA:	0x56
TABLE TAG SLAVE ADDR	0X20
DATA: SLAVE ADDR	0x18
DATA: (RESERVED)	0x00
DATA: (RESERVED)	0x00
DATA: TABLE-TAG: REGISTER DEF	0x48
DATA: REGISTER ID	0x00
DATA: BYTE COUNT	0x04
DATA:	0x12
DATA:	0x34
DATA:	0x56
DATA:	0x78
TABLE-TAG: END	0x80

PICkit™ Serial Analyzer User's Guide

11.7.2 Configuration / Status

11.7.2.1 CONTROL_BLOCK

TABLE 11-27:

BYTE	BIT	DESCRIPTION
COMM: I²C		
16	7:0	Mode: 0=default 1=interactive 2=automatic (register table)
17	7:0	bit flags – EVENT MARKERS
	0	1= event marker enable: ADDR RX
	1	1= event marker enable: DATA RX
	2	1= event marker enable: DATA TX
	3	1= event marker enable: ACK RX
	4	1= event marker enable: NACK RX
	5	1= event marker enable: REG READ
	6	1= event marker enable: REG WRITE
	7	1= event marker enable: status error
18	7:0	
	0	1= event marker enable: START (reserved – not implemented)
	1	1= event marker enable: STOP
	2	1= event marker enable: DATA RQ
	3	
	4	
	5	
	6	
	7	
19	7:0	
20	7:0	MODE 0 read data (bytes 1 thru N)
21	7:0	MODE 0 read data (byte 0)
22	7:0	SLAVE ADDRESS
23	7:0	SLAVE ADDRESS MASK

11.7.2.2 STATUS_BLOCK

TABLE 11-28:

BYTE	BIT	DESCRIPTION
		COMM: I ² Cs
12	7:0	bit flags: error status
	0	Slave profile malformed (mode 2)
	1	
	2	
	3	
	4	
	5	SSPOV
	6	WCOL
	7	composite error
13	7:0	bit flags: info status
	0	SCK line state (0 1)
	1	SDA line state (0 1)
	2	
	3	
	4	
	5	
	6	
	7	
14	7:0	
15	7:0	
16	7:0	
17	7:0	
18	7:0	
19	7:0	

PICKit™ Serial Analyzer User's Guide

11.7.3 Communication Tags

11.7.3.1 TAG: SCMND:

TABLE 11-29:

TAG / SCMND	LEN	NAME	DESCRIPTION
0xC0	1	I ² CS_INIT	initialize I ² C controller
			0 TAG
0xC1	N+2	I ² CS_SEND_BYTES (mode 1)	send/provide 'read' bytes
			0 TAG
			1 byte count (N)
			2 data
			N+1 data
0xC2	N+3	I ² CS_REGDATA_LOAD (mode 2)	write/load register definition
			0 TAG
			1 Slave address
			2 Register ID
			3 Byte count(N)
			4 Data
0xC3	3	I ² CS_REGDATA_RETURN (mode 2)	read/return register definition
			0 TAG
			1 Slave Address
			2 Register ID

11.7.3.2 TAG: SDATA:

TABLE 11-30:

TAG / SDATA	LEN	NAME	DESCRIPTION	
0xC0	2	I2CS_EVENT_ADDR	ADDRESS byte received	
			0	SDATA-TAG
			1	data (address)
0xC1	2	I2CS_EVENT_DATA_RX	DATA byte received (Master write)	
			0	SDATA-TAG
			1	data
0xC2	2	I2CS_EVENT_DATA_TX	DATA byte sent (Master read)	
			0	SDATA-TAG
			1	data
0xC3	1	I2CS_EVENT_ACK_RX	ACK received	
			0	SDATA-TAG
0xC4	1	I2CS_EVENT_NACK_RX	NACK received	
			0	SDATA-TAG
0xC5	1	I2CS_EVENT_START	START bit received	
			0	SDATA-TAG
0xC6	1	I2CS_EVENT_STOP	STOP bit received	
			0	SDATA-TAG
0xC7	2	I2CS_EVENT_STATUS_ER R	Status error	
			0	SDATA-TAG
			1	error status byte
0xC8	1	I2CS_DATA_RQ (mode 1)	DATA request (master read)	
			0	SDATA-TAG
0xC9	1	I2CS_EVENT_REG_READ (mode 2)	REG READ	
			0	SDATA-TAG
			1	Slave Address
			2	Register ID
0xCA	1	I2CS_EVENT_REG_WRIT E (mode 2)	REG WRITE	
			0	SDATA-TAG
			1	Slave Address
			2	Register ID
0xCB	N+3	I2CS_REGDATA (mode 2)	register data	
			0	TAG
			1	Slave Address
			2	Register ID
			3	Byte count = N
			4	Data
			N+3	Data

PICKit™ Serial Analyzer User's Guide

11.8 SPI COMMUNICATIONS

SPI is a 4-wire serial protocol that uses data in, data out, clock and Chip Select pins. It is a very basic protocol using a clock edge to capture data. The CONTROL_BLOCK is used to enable SPI event markers, set the bit rate and configure transaction/protocol options.

TABLE 11-31: CONNECTOR PINOUT IN SPI MODE

Pin	Description
1	CS
2	+V
3	GND
4	SDI
5	SCK
6	SDO

TABLE 11-32: SPI CONTROL BLOCK

Byte	Bit	Description
COMM: SPI		
16	7:0	Bit flags
	0	1 = Event marker enable: read byte
	1	1 = Event marker enable: write byte
	2	1 = Event marker enable: status error
	3	1 =
	4	1 =
	5	1 =
	6	1 =
17	7:0	SPI Configuration bits
	0	1 = SMP (sample phase)
	1	1 = CKE (clock edge select)
	2	1 = CKP (clock polarity)
	3	1 = DAOD (disable auto output disable on data input)
	4	n/a
	5	n/a
	6	n/a
7	CSPOL (Chip Select Polarity), 0=LO TRUE, 1=HI TRUE ⁽¹⁾	
18	7:0	
19	7:0	
20	7:0	
21	7:0	
22	7:0	BIT RATE: Pre-scale code
23	7:0	BIT RATE: Scaling code

Note 1: CSPOL determines the 'active' state of CS,
 if CSPOL=0, CS is active low
 if CSPOL=1, CS is active high
 upon initialization, CS is set 'inactive'

TABLE 11-33: BIT RATE CODES

Fosc	Pre-Scale Code	Pre-Scale Value	Scale Code	Scale Value	Total Scale Value	Bit Rate
20 MHz	0x00	8	0x00	1	8	2.500 MHz
20 MHz	0x00	8	0xFF	256	2048	9.766 kHz
20 MHz	0x01	32	0x00	1	32	0.625 MHz
20 MHz	0x01	32	0xFF	256	8192	2.441 kHz
20 MHz	0x02	128	0x00	1	128	0.156 MHz
20 MHz	0x02	128	0xFF	256	32768	0.610 kHz

TABLE 11-34: SPI STATUS BLOCK

Byte	Bit	Description
COMM: SPI		
12	7:0	Bit flags: error status
	0	WCOL error
	1	SSPOV error
	2	
	3	
	4	
	5	
	6	
	7	Composite error
13	7:0	SPI Configuration bits
	0	1 = SMP (sample phase)
	1	1 = CKE (clock edge select)
	2	1 = CKP (clock polarity)
	3	1 = AOD (auto output disable on data input)
	4	1 = SS
	5	1 =
	6	1 = SLAVE
	7	1 = MASTER
14	7:0	
15	7:0	
16	7:0	
17	7:0	
18	7:0	BIT RATE: Pre-scale (ref: section: x.x.x)
19	7:0	BIT RATE: Scaling (ref: section: x.x.x)

PICKit™ Serial Analyzer User's Guide

TABLE 11-35: SPI 'CMD' TAG BYTES

TAG/CCMD	LEN	Name	Description	
0x83	3	SPI_BITRATE	Set bit rate	
			0	TAG
			1	scaler
			2	pre-scaler (ref section x.x.x)
0x84	2	SPI_DATAIO_IN	Input data	
			0	TAG
			1	byte count (N)
0x85	N+2	SPI_DATAIO_OUT	Output data	
			0	TAG
			1	byte count (N)
			2	data
			N+1	data
0x86	N+2	SPI_DATAIO_INOUT	Output data	
			0	TAG
			1	byte count (N)
			2	data
			N+1	data
0x87	1	SPI_SDO_IN	Set SDO pin to INPUT (tri-state)	
			0	TAG
0x88	1	SPI_SDO_OUT	Set SDO pin to OUTPUT	
			0	TAG
0x8A	1	SPI_INIT	Initialize SPI controller per CONTROL BLOCK	
			0	TAG
0x8B	1	SPI_CS_ON	Assert CS ⁽¹⁾	
			0	TAG
0x8C	1	SPI_CS_OFF	De-assert CS ⁽¹⁾	
			0	TAG

Note 1: CS polarity is determined by CONTROL BLOCK configuration bit CSPOL
 CSPOL=0, asserting sets CS=0, de-asserting sets CS=1
 CSPOL=1, asserting sets CS=1, de-asserting sets CS=0

TABLE 11-36: SPI 'DATA' TAG BYTES

TAG/CDATA	LEN	Name	Description	
0x80	2	SPI_EVENT_BYTE_TX	BYTE transmit	
			0	TAG
			1	data
0x81	2	SPI_EVENT_BYTE_RX	BYTE transmit	
			0	TAG
			1	data
0x82	2	SPI_EVENT_STATUS_ERR	Status error	
			0	TAG
			1	error byte

11.9 USART COMMUNICATIONS

Universal Synchronous Asynchronous Receive Transmit (USART) protocol is a standard 2-wire serial communication. In Asynchronous mode, there is a transmit line and a receive line. In Synchronous mode, the transmit line becomes the clock line and the receive line becomes the bidirectional data line. In Asynchronous mode, 8 bits are framed by a Start and Stop bit. A ninth bit can be included to use as a parity bit.

To configure the protocol, set the clock polarity to rising edge or falling edge if using Synchronous mode. Select the baud rate according to the BRG table.

TABLE 11-37: CONNECTOR PINOUT IN USART ASYNCHRONOUS MODE

Pin	Description
1	TX
2	+V
3	GND
4	—
5	—
6	RX

TABLE 11-38: CONNECTOR PINOUT IN USART SYNCHRONOUS MODE

Pin	Description
1	Clock
2	+V
3	GND
4	—
5	—
6	Data

PICkit™ Serial Analyzer User's Guide

TABLE 11-39: USART CONTROL BLOCK

Byte	Bit	Description
COMM: USART		
16	7:0	n/a
17	7:0	
	0	1 = clock polarity (df1t) 1 = hi-lo at start bit cell, 0 = lo-hi (USART-SM only, ignored otherwise)
	1	1 = n/a
	2	1 = Async receive disable (USART-A only, ignored otherwise)
	3	1 = n/a
	4	1 = n/a
	5	1 = n/a
	6	1 = n/a
18	7:0	
	0	1 = Event marker enable: read byte
	1	1 = Event marker enable: write byte
	2	1 = Event marker enable: status error
	3	1 = Event marker enable: break tx
	4	1 = n/a
	5	1 = n/a
	6	1 = n/a
7	1 = n/a	
19	7:0	n/a
20	7:0	n/a
21	7:0	bit flags
	0	AUX1 – default state (0 1)
	1	AUX2 – default state (0 1)
	2	AUX1 – direction (1=IN, 0=OUT)
	3	AUX2 – direction (1=IN, 0=OUT)
	4	n/a
	5	n/a
	6	n/a
7	n/a	
22	7:0	BRG (BAUD) default (LSB)
23	7:0	BRG (BAUD) default (MSB)

TABLE 11-40: BRG BAUD RATE TABLE

BAUD = Fosc/(4*(BRG + 1))			
BAUD	BRG	ACTUAL	ERR
300	16666	300.0	0.00%
1200	4166	1199.9	-0.01%
4800	1041	4798.5	-0.03%
9600	520	9596.9	-0.03%
19200	259	19230.8	0.16%
28800	173	28735.6	-0.22%
57600	86	57471.3	-0.22%
115200	42	116279.1	0.94%

TABLE 11-41: USART STATUS BLOCK

Byte	Bit	Description
COMM: USART		
12	7:0	Bit flags: error status
	0	FERR – framing error
	1	OERR – overrun error
	2	INIT error (bad “mode”)
	3	n/a
	4	n/a
	5	n/a
	6	n/a
	7	<i>Composite error</i>
13	7:0	
14	7:0	
	0	1 = clock polarity (df1t) 1 = hi-lo at start bit cell, 0 = lo-hi (USART-SM only, ignored otherwise)
	1	1 = n/a
	2	1 = Async receive disabled (USART-A only, ignored otherwise)
	3	n/a
	4	n/a
	5	n/a
	6	n/a
	7	n/a
15	7:0	
16	7:0	
	0	AUX1 – state (0 1)
	1	AUX2 – state (0 1)
	2	AUX1 – direction (1=IN, 0=OUT)
	3	AUX2 – direction (1=IN, 0=OUT)
	4	n/a
	5	n/a
	6	n/a
	7	n/a
17	7:0	n/a
18	7:0	BRG (BAUD) default (LSB)
19	7:0	BRG (BAUD) default (MSB)

PICkit™ Serial Analyzer User's Guide

TABLE 11-42: USART 'CMD' TAG BYTES

TAG/ CCMD	LEN	Name	Description
0x80	1	USART_INIT	Initialize USART controller
			0 TAG
0x82	N+2	USART_DATA_XMT	Transmit data
			0 TAG
			1 Byte count = N
			2 Data
			N+1 Data
0x83	2	USART_DATA_SRCV (USART-SM only)	Receive data
			0 TAG
			1 Byte count = N
0x84	1	USART_DATA_ARCV_ENABLE (USART-A only)	Enable receive data monitor
			0 TAG
0x85	1	USART_DATA_ARCV_DISABLE (USART-A only)	Disable receive data monitor
			0 TAG
0x86	1	USART_BREAK_XMT (USART-A only)	send BREAK
			0 TAG
0x87	2	USART_BREAK_DATA_XMT (USART-A only)	send BREAK then DATA byte
			0 TAG
			1 DATA byte (TYP: 0x55)
0x88	3	USART_BAUD	set BAUD rate
			0 TAG
			1 BAUD value (LSB)
			2 BAUD value (MSB) (Reference section x x x)
0x89	1	USART_SCKP_SET (USART-SM only)	set "CLOCK POLARITY" bit
			0 TAG
0x8A	1	USART_SCKP_RST (USART-SM only)	reset "CLOCK POLARITY" bit
			0 TAG
0x90	1	USART_AUX1_RST	AUX1: 0
			0 TAG
0x91	1	USART_AUX1_SET	AUX1: 1
			0 TAG
0x92	1	USART_AUX1_OUT	AUX1 direction: OUTPUT
			0 TAG
0x93	1	USART_AUX1_IN	AUX1 direction: INPUT
			0 TAG
0x94	1	USART_AUX1_WAIT_0	wait AUX1 == 0
			0 TAG
0x95	1	USART_AUX1_WAIT_1	wait AUX1 == 1
			0 TAG
0x96	1	USART_AUX2_RST	AUX2: 0
			0 TAG

TABLE 11-42: USART 'CMD' TAG BYTES (CONTINUED)

0x97	1	USART_AUX2_SET	AUX2: 1	
			0	TAG
0x98	1	USART_AUX2_OUT	AUX2 direction: OUTPUT	
			0	TAG
0x99	1	USART_AUX2_IN	AUX2 direction: INPUT	
			0	TAG
0x9A	1	USART_AUX2_WAIT_0	wait AUX2 == 0	
			0	TAG
0x9B	1	USART_AUX2_WAIT_1	wait AUX2 == 1	
			0	TAG

TABLE 11-43: USART 'DATA' TAG BYTES

TAG/ CDATA	LEN	Name	Description	
0x80	2	USART_EVENT_BYTE_TX	BYTE transmit	
			0	TAG
			1	data
0x81	2	USART_EVENT_BYTE_RX	BYTE received	
			0	TAG
			1	data
0x82	2	USART_EVENT_STATUS_ERR	Status error	
			0	TAG
			1	error byte
0x83	1	USART_EVENT_BREAK_TX	BREAK transmitted	
			0	TAG

11.10 LIN COMMUNICATIONS

11.10.1 Introduction

The PKSA 'LIN' support is confined to MASTER and MONITOR functionality. As a MASTER, the PKSA accepts instruction from the HOST in the form of script TAG(s). Execution of the script results in the PKSA transmitting on the LIN bus as a MASTER. In practice, the transmission would constitute a LIN frame header or an entire LIN frame i.e. message.

As a LIN bus MONITOR, the PKSA can detect and report all events / transactions on the bus generated by other devices on the bus or generated by the PKSA MASTER module. The 'event marker' TAG(s) are used to report bus activity in its variety of forms but must be enabled, individually, using CONTROL_BLOCK configuration bits as seen below.

11.10.2 Configuration / Status

11.10.2.1 CONTROL_BLOCK

TABLE 11-44:

BYTE	BIT	DESCRIPTION
COMM: LIN		
16	7:0	bit flags
	0	1= n/a
	1	1= n/a
	2	1= n/a
	3	1= set LIN adapter chip select line HI
	4	1= n/a
	5	1= n/a
	6	1= receive enable
	7	1= enable auto baud detect / set
17	7:0	bit flags
	0	1= event marker enable: data RX
	1	1= event marker enable: data TX
	2	1= event marker enable: break TX
	3	1= event marker enable: break RX
	4	1= event marker enable: auto baud
	5	n/a
	6	n/a
	7	1= event marker enable: status error
18	7:0	n/a
19	7:0	n/a
20	7:0	n/a
21	7:0	n/a
22	7:0	BAUD rate (BRG code) default (LSB)
23	7:0	BAUD rate (BRG code) default (MSB)

11.10.2.2 STATUS_BLOCK

TABLE 11-45:

BYTE	BIT	DESCRIPTION
COMM: LIN		
12	7:0	bit flags: error
	0	FERR – framing error
	1	OERR – overrun error
	2	FAULT – from LIN bus adapter
	3	AUTOBAUD detect error (e.g. overflow)
	4	n/a
	5	n/a
	6	INIT error
	7	composite error
13	7:0	bit flags: status
	0	PIN state: FAULT
	1	PIN state: CS
	2	AUTOBAUD detect in progress
	3	n/a
	4	n/a
	5	n/a
	6	n/a
	7	n/a
14	7:0	n/a
15	7:0	n/a
16	7:0	n/a
17	7:0	n/a
18	7:0	currently active BAUD rate (BRG code) (LSB)
19	7:0	currently active BAUD rate (BRG code) (MSB)

11.10.2.3 CONFIGURATION CONSIDERATIONS

The PKSA can be configured to operate as LIN bus MASTER, MONITOR or both.

MASTER

To configure the PKSA to perform as LIN bus MASTER, the HOST must write the CONTROL_BLOCK to specify:

1. default BAUD rate
2. LIN adapter 'chip select' line state

Optionally,

1. set 'receive enable' configuration bit to allow the PKSA to receive RX data and report it to the HOST
2. enable desired event markers

MONITOR

To configure the PKSA to perform as LIN bus MONITOR, the HOST must write the CONTROL_BLOCK to specify:

1. LIN adapter 'chip select' line state
2. set 'receive enable' configuration bit to allow the PKSA to receive RX data and

report it to the HOST

3. set 'auto baud enable' configuration bit to allow PKSA to track baud rate of bus transactions

Optionally,

1. enable desired combination of event markers

11.10.3 Communication Tags

11.10.3.1 TAG: SCMND:

TAG commands and data sent to the PICkit Serial Analyzer.

TABLE 11-46:

TAG / SCMND	LEN	NAME	DESCRIPTION
0x80	1	LIN_INIT	initialize USART controller
			0 TAG
0x81	N+2	LIN_XMT_DATA ⁽¹⁾	transmit DATA byte(s)
			0 TAG
			1 Byte count = N
			2 Data
			N+1 Data
0x82	1	LIN_XMT_BRK	transmit BREAK
			0 TAG
0x83	N+2	LIN_XMT_BRK_DATA ⁽¹⁾	transmit BREAK then DATA bytes (the data must include SYNC char)
			0 TAG
			1 Byte count = N
			2 Data (SYNC character)
			N+1 Data
0x84	N+2	LIN_XMT_FRAME ⁽¹⁾	transmit DATA byte(s) (implied leading BREAK & SYNC)
			0 TAG
			1 Byte count = N
			2 Data
			N+1 Data
0x85	3	LIN_BAUD_SET	set BAUD rate
			0 TAG
			1 BAUD code (LSB)
			2 BAUD code (MSB)

Note 1: Data bytes (i.e. 'frame' to be published) must include the appropriate CHECKSUM – 'enhanced' CHECKSUM if frame is destined for a LIN 2.0 slave. The PKSA has no comprehension of data content or 'format' – data bytes are simply transmitted verbatim.

11.10.3.2 TAG: SDATA:

TAG commands and data returned from the PICKit Serial Analyzer.

TABLE 11-47:

TAG / SDATA	LEN	NAME	DESCRIPTION
0x80	2	LIN_EVENT_BYTE_RX	BYTE received
			0 TAG
			1 data
0x81	2	LIN_EVENT_BYTE_TX	BYTE transmitted
			0 TAG
			1 data
0x82	2	LIN_EVENT_STATUS_ERR	status error
			0 TAG
			1 error byte
0x83	1	LIN_EVENT_BREAK_RX	BREAK received
			0 TAG
0x84	1	LIN_EVENT_BREAK_TX	BREAK transmitted
			0 TAG
0x85	3	LIN_EVENT_AUTOBAUD	AUTOBAUD event
			0 TAG
			1 BAUD CODE (LSB)
			2 BAUD CODE (MSB)

Those TAGs (above) identified as 'EVENT', generally mark the occurrence of LIN bus activity. As with all other supported PKSA serial protocols, the 'EVENT' TAGs are generated when/as are detected in real time and queued in a circular buffer for transmission to the HOST. Each TAG can be individually enabled/disabled using the appropriate configuration bit in the CONTROL BLOCK.

11.10.4 Parameter Detail

11.10.4.1 BAUD

The PKSA BAUD rate is configured by a 2-byte code as detailed in the table(s) below. Upon initialization, the PKSA is configured with the 'default' BAUD specified in the CONTROL BLOCK (bytes 22 & 23) as documented in the table above. Additionally, the BAUD rate can be changed in the context of script execution (real time) using the appropriate script/SCMND TAG. At all times the STATUS BLOCK reflects the currently active BAUD rate. This BAUD rate applies to all data transmitted by the PKSA.

AUTOBAUD

The PKSA-LIN 'monitor' function has the flexibility to auto baud detect and modify the PKSA hardware configuration to receive 'messages' from the LIN bus at the BAUD rate being transmitted. This 'flexibility' must be enabled using the configuration bit in the CONTROL BLOCK (byte 16 bit 7). The HOST can be apprised of the 'auto baud' detect action through an event marker TAG.

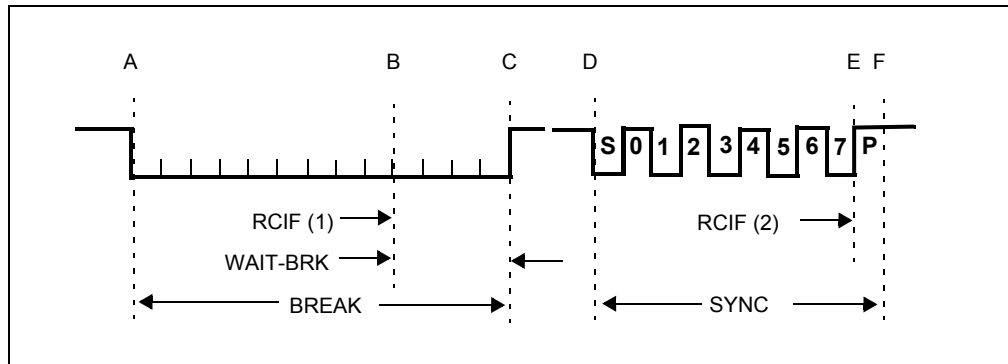
The AUTOBAUD operation depends on receiving a 'BREAK' (13 bits of 0s) followed by a 'SYNC' (byte 0x55). Referring to the timing diagram below – the BREAK begins at 'A'. At 'B' the PKSA expected a STOP bit – the USART interrupts the PKSA (RCIF – receive interrupt flag) and sets FERR (framing error). The PKSA interprets (1) DATA=0 and (2) FERR (framing error) as having received a BREAK; therefore, the FERR is automati-

cally reset and not reported to the host. If AUTOBAUD detection is enabled, the PKSA will wait a maximum of 'WAIT-BRK' for the BREAK to complete at 'C'. The duration of WAIT-BRK must be set to accommodate the slowest BAUD rate supported (6 msec was chosen to support 1000 BAUD +/- 50%). If the end of BREAK is not seen in that time, the AUTOBAUD ERROR bit is set in the STATUS_BLOCK and the operation is aborted. If the end of BREAK is seen, the AUTOBAUD mechanism is enabled and the BAUD rate of the SYNC character is measured. At the second interrupt (RCIF (2)) at 'E', the operation is complete and the new/measured BAUD rate is used for subsequent transactions. If the AUTOBAUD measurement experienced an overflow, the AUTOBAUD ERROR bit is set in the STATUS_BLOCK and the default BAUD rate from the CONTROL_BLOCK is reloaded into the USART. In either case, the resultant BAUD rate is reflected in the STATUS_BLOCK. After the BREAK, the PKSA will wait indefinitely for the SYNC character to begin with the USART receiver disabled.

WARNING

The SYNC character is not actually 'received' by the PKSA during an AUTOBAUD operation; therefore, it is not reflected/reported to the HOST. The EVENT MARKER sequence would be BREAK_RX and AUTOBAUD followed by BAUD_L, BAUD_H. If AUTOBAUD was not enabled, the EVENT MARKER sequence would be BREAK_RX then BYTE_RX followed by 0x55.

FIGURE 11-5:



PICKit™ Serial Analyzer Firmware

LIN specifications limit the BAUD rate to 20k (Reference the following table for examples).

$$\text{BAUD} = \text{FOSC} / (4 * (\text{BRG_CODE} + 1))$$

where: FOSC = 20 MHz

TABLE 11-48:

BAUD	CODE	ACTUAL	ERROR
100	49999	100.0	0.00%
200	24999	200.0	0.00%
300	16666	300.0	0.00%
400	12499	400.0	0.00%
500	9999	500.0	0.00%
600	8332	600.0	0.00%
700	7142	700.0	0.00%
800	6249	800.0	0.00%
900	5555	899.9	-0.01%
1000	4999	1000.0	0.00%
2000	2499	2000.0	0.00%
3000	1666	2999.4	-0.02%
4000	1249	4000.0	0.00%
5000	999	5000.0	0.00%
6000	832	6002.4	0.04%
7000	713	7002.8	0.04%
8000	624	8000.0	0.00%
9000	555	8992.8	-0.08%
10000	499	10000.0	0.00%
11000	454	10989.0	-0.10%
12000	416	11990.4	-0.08%
13000	384	12987.0	-0.10%
14000	356	14005.6	0.04%
15000	332	15015.0	0.10%
16000	312	15974.4	-0.16%
17000	293	17006.8	0.04%
18000	277	17985.6	-0.08%
19000	262	19011.4	0.06%
20000	249	20000.0	0.00%

PICKit™ Serial Analyzer User's Guide

NOTES:

Chapter 12. PICKit™ Serial Analyzer DLL

12.1 INTRODUCTION

Custom software programs can be created by accessing the Dynamically Linked Library (DLL), PICKitS.dll, with the software language of your choice. All of the functionality to create tag byte scripts that will be converted to protocol scripts is built into the DLL and is very easy to use. Any programming language that can access functions from a DLL can be used, however, a Microsoft .NET framework language is recommended.

12.2 OVERVIEW

The name of the DLL is PickitS.dll and is included in the PICKit Serial Analyzer installation. The functions used to create custom GUI apps are Divided into logical classes inside the DLL. The classes are listed below:

- I2C Master (class PICKitS.I2CM)
- I2C Slave (class PICKitS.I2CS)
- SPI Master (class PICKitS.SPIM)
- Microwire Master (class PICKitS.MicrowireM)
- USART (class PICKitS.USART)
- LIN (class PICKitS.LIN)
- Device (class PICKitS.Device)
- USBRead (Class PICKitS.USBRead)

Applications call functions from class 'Device' for initialization and cleanup, and from a specific protocol class for input and output.

Example projects (written in Visual Basic .NET Express – a free download from Microsoft) and detailed documentation for each protocol are available for download from the PICKit Serial Website.

PICKit™ Serial Analyzer User's Guide

Chapter 13. Troubleshooting

13.1 INTRODUCTION

This chapter describes questions and answers to common problems associated with using the PICKit™ Serial Analyzer and how to resolve them.

13.2 FREQUENTLY ASKED QUESTIONS

PICKit™ Serial Analyzer could not be found

Question

I am receiving the error message, "PICKit™ Serial Analyzer could not be found" in the Transactions window, but the PICKit™ Serial Analyzer is plugged in. What is wrong?

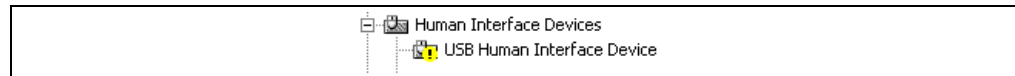
Answer

Open the Windows® operating system Device Manger by clicking on *Control Panel > System*, selecting the Hardware tab and clicking on **Device Manger** button. Check if there is an error displayed under Human Interface Devices as shown in Figure 13-1.

If an error is displayed, try unplugging and the re-plugging the USB cable until the error goes away (this may take 3 or 4 tries).

If the error persists, try another USB port or hub. Try plugging the PICKit™ Serial Analyzer into another computer to verify that the USB port is working.

FIGURE 13-1: WINDOWS® DEVICE MANAGER



Current Limit Exceeded

Question

I received the error message "USB Hub Current Limit Exceeded" from the Windows® operating system. What is wrong?

Answer

The USB port current limit is set to 100 mA. If the target device plus PICKit™ Serial Analyzer exceeds this current limit, the USB port will shut down. Check for shorts. The target device can be externally powered if more power is needed.

Microsoft® Windows® 98 SE

Question

After plugging the PICKit™ Serial Analyzer into the USB port, Windows® 98 SE asks for a driver. Where is the driver?

Answer

PICKit™ Serial Analyzer uses the Human Interface Device (HID) driver included with the Windows® Operating System. When Windows® 98 SE prompts for a driver, select "Search for the best driver for your device." Then select the check box next to "Microsoft Windows Update" and click Next. Windows will automatically install the appropriate driver. Do not use Microchip's MPLAB® ICD 2 USB driver.

Microsoft® Windows® 95/98/NT

Question

Can I run on Windows® 95/98/NT?

Answer

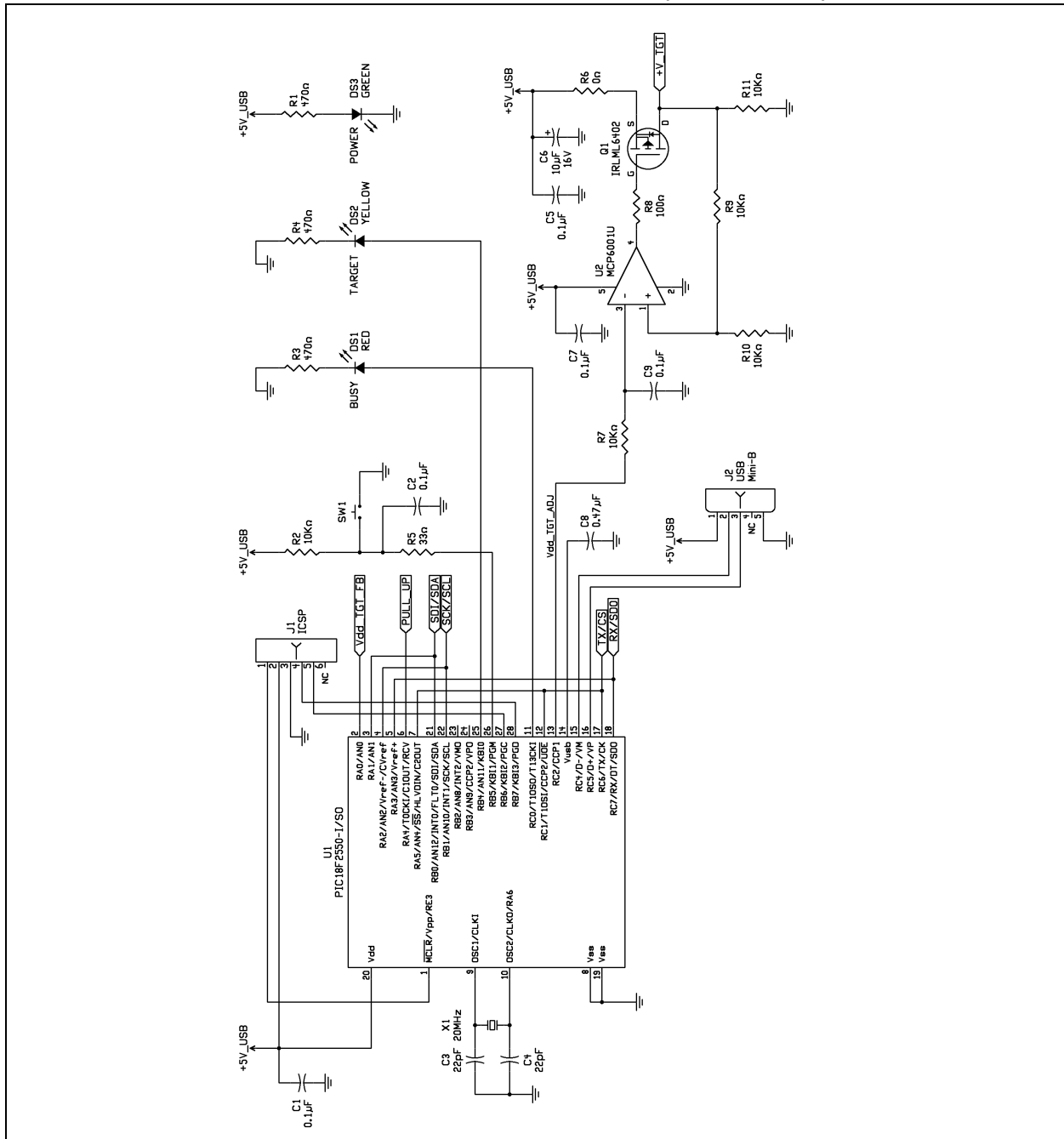
No. These operating systems either do not support USB or have drivers that are not compatible.

Appendix A. PICKit Serial Analyzer Schematics

A.1 INTRODUCTION

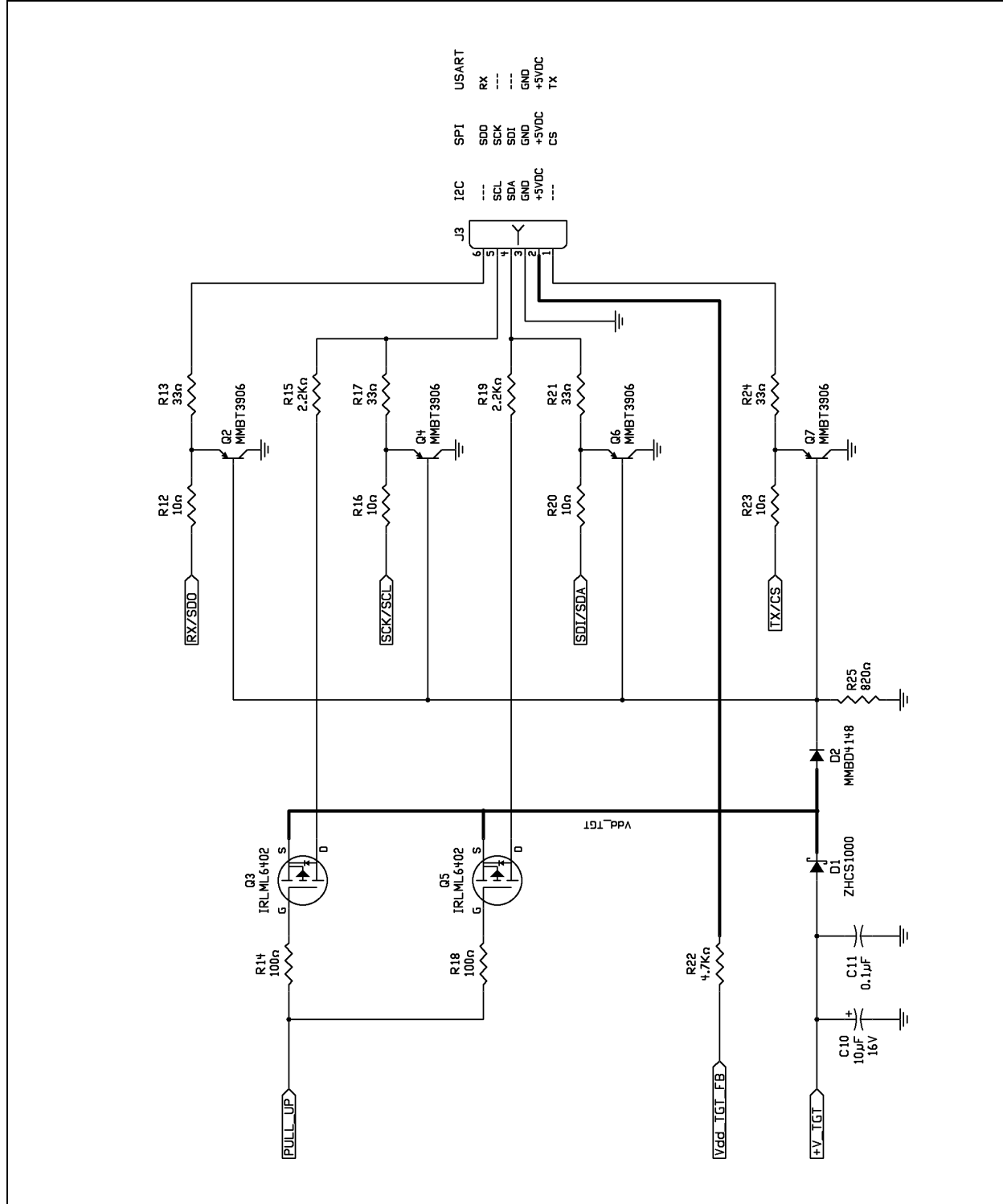
This appendix contains the PICKit Serial Management hardware diagrams.

FIGURE A-1: PICKit™ SERIAL ANALYZER SCHEMATIC (SHEET 1 OF 2)



PICKit™ Serial Analyzer User's Guide

FIGURE A-2: PICKit™ SERIAL ANALYZER SCHEMATIC (SHEET 2 OF 2)



PICKIT Serial Analyzer Schematics

FIGURE A-3: SILKSCREEN

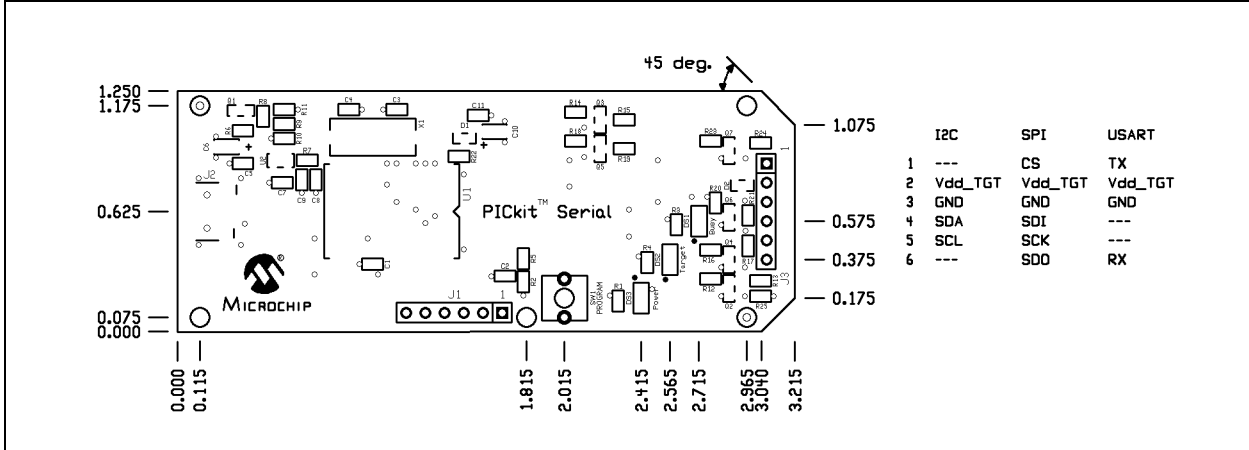


FIGURE A-4: TOP COPPER

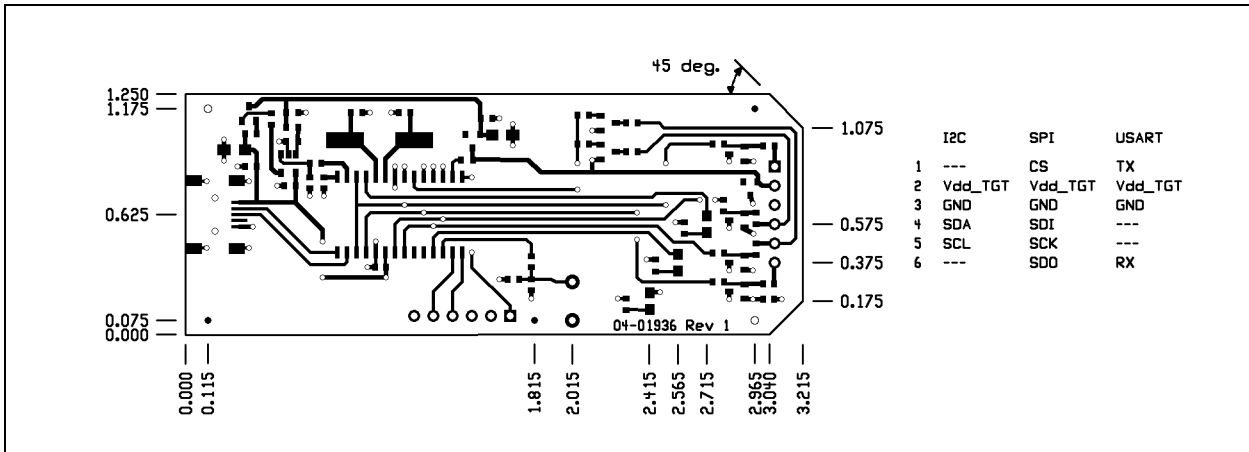
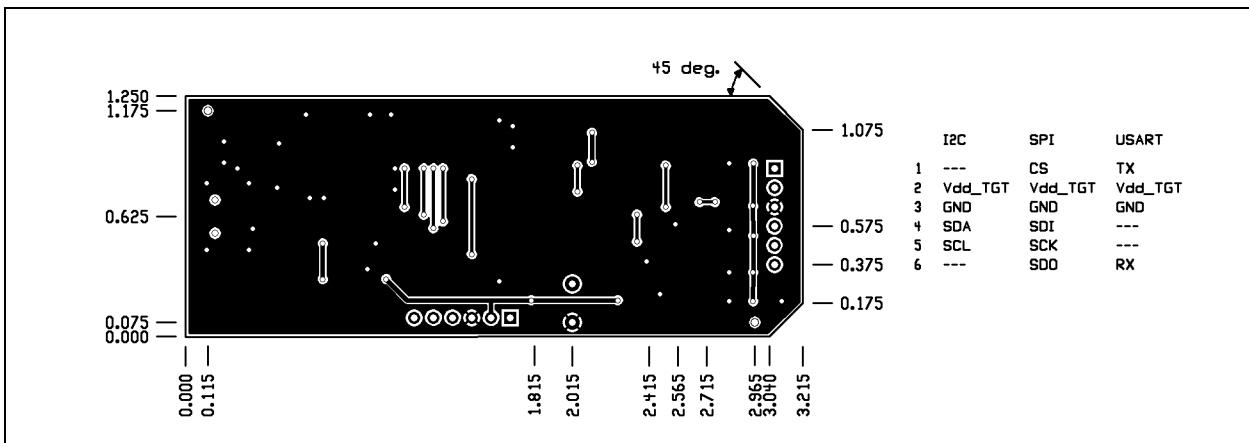


FIGURE A-5: BOTTOM COPPER



PICKit™ Serial Analyzer User's Guide

NOTES:

Appendix B. 28-Pin Demo Board I²C™ Demonstration Firmware

B.1 INTRODUCTION

The 28-Pin Demo Board I²C™ demonstration firmware communicates with the PICKit™ Serial Analyzer using the I²C serial protocol. The PICKit Serial Analyzer will be the I²C Master and the 28-Pin Demo Board will be the I²C Slave device. The 28-Pin Demo board is programmed to emulate an I²C Real-Time Clock (RTC) and Serial EEPROM.

B.2 HIGHLIGHTS

This chapter discusses:

- Hardware
- Firmware
- I²C Communications
- Slave Devices
- Functions

B.3 HARDWARE

The 28-Pin Demo Board (DM164120-3) is populated with and configured for:

- PIC16F886 configured using the 8 MHz internal clock
- 32 kHz Tuning Fork Crystal connected to Timer1 Low-power Oscillator
- Four LEDs (DS1 through DS4) connected to RB0 through RB3
- Potentiometer (RP1) connected to RA0/AN0
- Push button (SW1) connected to RE3/MCLR

For more information about the 28-Pin Demo Board hardware, see the 28-Pin Demo Board User's Guide (DS41301).

B.4 FIRMWARE

The demo program source code and *.hex file can be found on the PICKit Serial CD-ROM at D:\28-pin Demo Board\Firmware\.

The firmware is structured as multiple 'modules' representing functional components. Each module contains, at minimum, an "initialization" routine (MODULE_INIT) and a "service" routine (MODULE_SVC). Each initialization routine is called from MAIN once on Reset. Each Interrupt Service Routine is called sequentially and continuously from the MAIN Idle loop. Interrupt Service Routine is provided for the I²C module only. All other modules are serviced in turn from the MAIN 'Idle loop'.

PICKit™ Serial Analyzer User's Guide

TABLE B-1: FIRMWARE MODULES

MODULE	DESCRIPTION
adc.asm	ADC service – measuring channel AN0, connected to potentiometer RP1, and post results to register in shared memory
device.asm	Basic device (microcontroller) configuration
ee_util.asm	EEPROM Read/Write routines
exec.asm	“Executive” feature set – provides functionality for test and demonstration of PICKit™ Serial Analyzer and 28-Pin Demo Board
i2c_slave_pkasd.asm	I ² C™ service for three slave devices: 0xA2: Real-Time Clock (RTC) 0xA8: EEPROM 0xAA: Executive (EXEC)
led.asm	LED management, set/reset LEDs per state register
main.asm	Initialization, Idle loop and Interrupt Service Routine context management
pski.asm	Monitor PICKit Serial Analyzer interface I/O pins – post results to state register
rtc.asm	I ² C™ device: real-time clock emulation maintains 16 locations for register-based I/O
timer0.asm	Utilizes TMR0 as a source for multiple “soft” timers
timer1.asm	Utilizes TMR1 with external 32 kHz crystal for RTC 1-second ‘ticks’

B.5 I²C COMMUNICATIONS

B.5.1 Overview

The 28-Pin Demo Board I²C™ Demo responds to three I²C slave addresses:

TABLE B-2: DEVICE SLAVE ADDRESSES

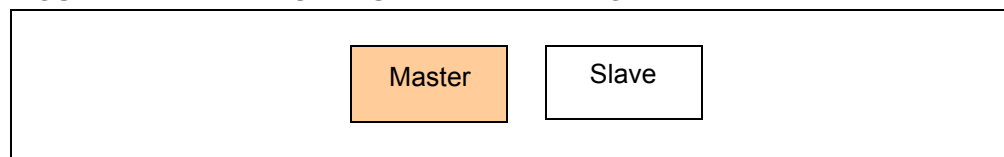
ADDR	NAME	DESCRIPTION
0xA2	RTC	Device emulation: Real-Time Clock
0xA8	EEPROM	Device emulation: EEPROM
0xAA	EXEC	Supervisory features

All devices (slave addresses) respond to the I²C protocols described below. Each slave device supports a fixed number of word addresses (see section **B.6 Slave Devices** below). The word address is automatically incremented during a transaction for sequential access of the device registers. If the word address is incremented beyond the end of the supported address range, the address will “wrap”.

B.5.2 Protocols

Figure B-1 shows the Master and Slave legend used for Figures B-2 through B-4.

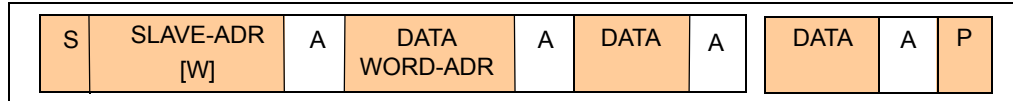
FIGURE B-1: MASTER/SLAVE DEVICE LEGEND



Write Byte(s) – This transaction is used to write one or more bytes. The maximum number of bytes is slave-dependant.

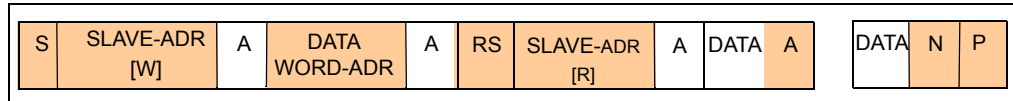
28-Pin Demo Board I²C™ Demonstration Firmware

FIGURE B-2: I²C™ WRITE BYTE(S)



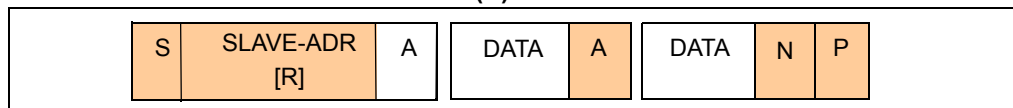
Read Byte(s) with Word Address – The word address is set to begin at a given value and incremented sequentially during the transaction.

FIGURE B-3: I²C™ READ BYTE(S) WITH WORD ADDRESS



Read Byte(s) without Word Address – The word address will continue in sequence from the previous transaction to the I²C slave. On Reset the device address for all three slave devices are reset to zero.

FIGURE B-4: I²C™ READ BYTE(S) WITHOUT WORD ADDRESS



B.6 SLAVE DEVICES

The 28-Pin Demo Board I²C™ Demonstration firmware responds to three I²C slave devices: RTC, Serial EEPROM, and Executive.

B.6.1 Slave Address: 0xA2 – Real-Time Clock Emulation

The 28-Pin Demo Board I²C™ Demonstration emulates an I²C Real-Time Clock (RTC). Table B-3 list the word addresses.

TABLE B-3: REAL-TIME CLOCK (RTC) WORD ADDRESSES

REG	NAME	DESCRIPTION
0x00	RTC_CONFIG_1	7: 0 6: 0 5: STOP – stop time function (0: RUN, 1: STOP) 4: 0 3: 0 2: 0 1: 0 0: 0
0x01	RTC_CONFIG_2	7: 0 6: 0 5: 0 4: 0 3: AF – Alarm flag 2: 0 1: AE – Alarm enable 0: 0
0x02	RTC_SECONDS	00-59 seconds, coding: BCD, bit 7: VL?
0x03	RTC_MINUTES	00-59 minutes, coding: BCD
0x04	RTC_HOURS	00-23 hours, coding: BCD
0x05	RTC_WEEKDAY	00-06 weekday
0x06	RTC_DAYS	01-31 day of the month
0x07	RTC_MONTHS	01-12 month of the year, coding: BCD

TABLE B-3: REAL-TIME CLOCK (RTC) WORD ADDRESSES (CONTINUED)

0x08	RTC_YEARS	00-99 year, coding: BCD
0x09	RTC_ALARM_MIN	00-59 minute of alarm, coding: BCD bit 7: enable
0x0A	RTC_ALARM_HOUR	00-23 hour of alarm, coding: BCD bit 7: enable
0x0B	RTC_ALARM_DAY	01-31 day of alarm, coding BCD bit 7: enable
0x0C	RTC_ALARM_WEEK	00-06 weekday of alarm bit 7: enable
0x0D	(not assigned)	
0x0E	(not assigned)	
0x0F	(not assigned)	

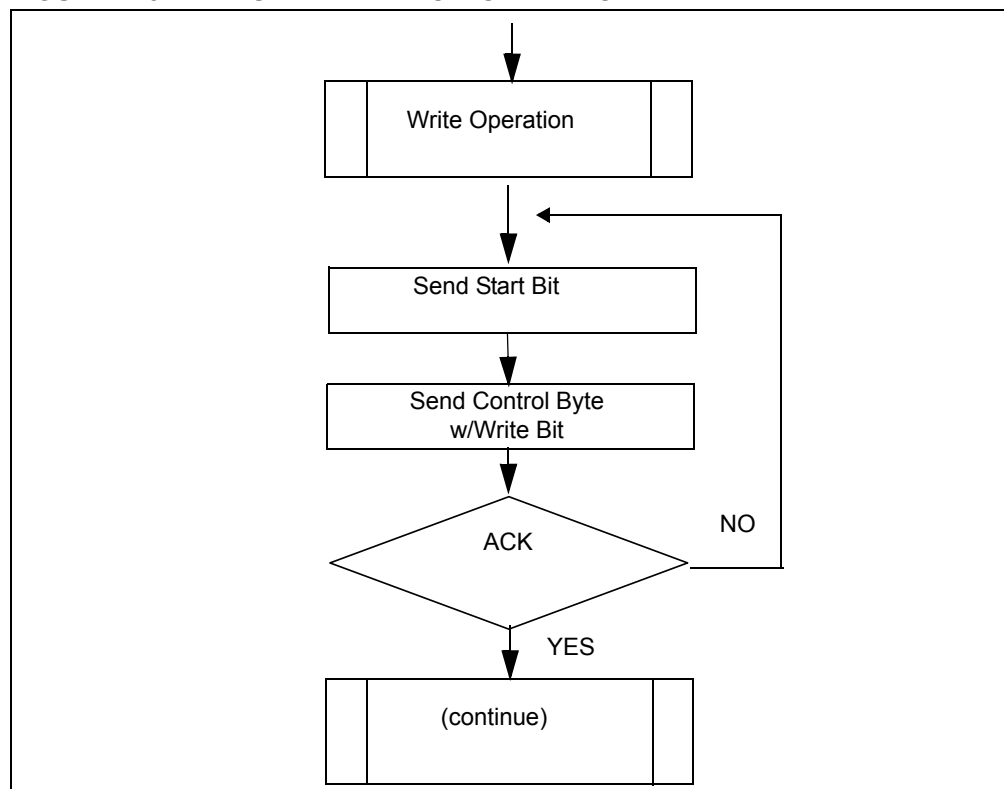
B.6.2 Slave Address: 0xA8 – EEPROM Emulation

The 28-Pin Demo Board I²C™ Demonstration emulates a serial EEPROM. Device emulation provides for 256 bytes. A newly-programmed 28-Pin Demo Board defaults the first 8 bytes to 0x00 to 0x07 sequentially. The value of the remaining bytes is undefined / unknown.

A Read operation is executed during the transaction.

The Write operation begins after the Stop bit is received. The firmware will not respond to I²C communications during the Write.

FIGURE B-5: SERIAL EEPROM OPERATION



B.6.3 Slave Address: 0xAA – Executive (EXEC)

The “Executive” I²C pseudo device provides features convenient for testing and demonstrating the PICkit™ Serial Analyzer and the 28-Pin Demo Board I²C™ firmware. The I²C word addresses are listed in Table B-4.

28-Pin Demo Board I²C™ Demonstration Firmware

TABLE B-4: TABLE B-4 EXECUTIVE (EXEC) WORD ADDRESSES

REG	NAME	DESCRIPTION
0x00	EXEC_STATE	State of executive state controller bit 7 == 1: force state controller to one of eight "entry points" specified by bits[2:0]
0x01	EXEC_ADC_CH0	ADC results of CHANNEL AN0 (potentiometer RP1)
0x02	EXEC_RTC_SECS	binary value representation of RTC "SECONDS"
0x03	EXEC_STATUS	state of communications connector pins: bit 0: PIN 1 (AUX1) bit 1: PIN 4 (SDA) bit 2: PIN 5 (SCL) bit 3: PIN 6 (AUX2) bits[5:4]: undefined bit 6: operation error bit 7: operation busy
0x04	EXEC_04	undefined (can be read/written with no operational effect)
0x05	EXEC_05	undefined (can be read/written with no operational effect)
0x06	EXEC_06	undefined (can be read/written with no operational effect)
0x07	EXEC_VERSION	Firmware version number

State Controller – The Executive state controller manages background functions. The HOST enacts a function by writing to EXEC_STATE with a valid "state entry value" (i.e., 0x80-0x87). Writing values outside this range will cause unknown results.

TABLE B-5: EXEC STATE ENTRY VALUES

VALUE	NAME	DESCRIPTION
0x80	EXEC_DISPLAY_PING_PONG	display PING-PONG LED pattern: DS1,2,3,4,3,2,1,2 ...
0x81	EXEC_TIMER1_TEST	operation: timer1 test
0x82	EXEC_DISPLAY_ADC	display of Most Significant 4 bits of ADC results measuring variable resistor
0x83	EXEC_DISPLAY_RTC	display of RTC SECONDS (binary) – 0-15
0x84	EXEC_DISPLAY_PKSI	display of state of communications connector pins: LED1: PIN 1 (AUX1) LED2: PIN 4 (SDA) LED3: PIN 5 (SCA) LED4: PIN 6 (AUX2)
0x85	EXEC_DISPLAY_RESET	display 'RESET' sequence
0x86	EXEC_DISPLAY_1SEC	display – blink all LEDs – 1 sec ON, 1 sec OFF, ...
0x87	EXEC_0x87	undefined

PICKit™ Serial Analyzer User's Guide

B.7 FUNCTIONS

B.7.1 Ping-Pong Display

The PING-PONG display is useful in testing the four LEDs which are illuminated individually and in sequence as seen below. The following pattern is executed at a 100-millisecond step interval.

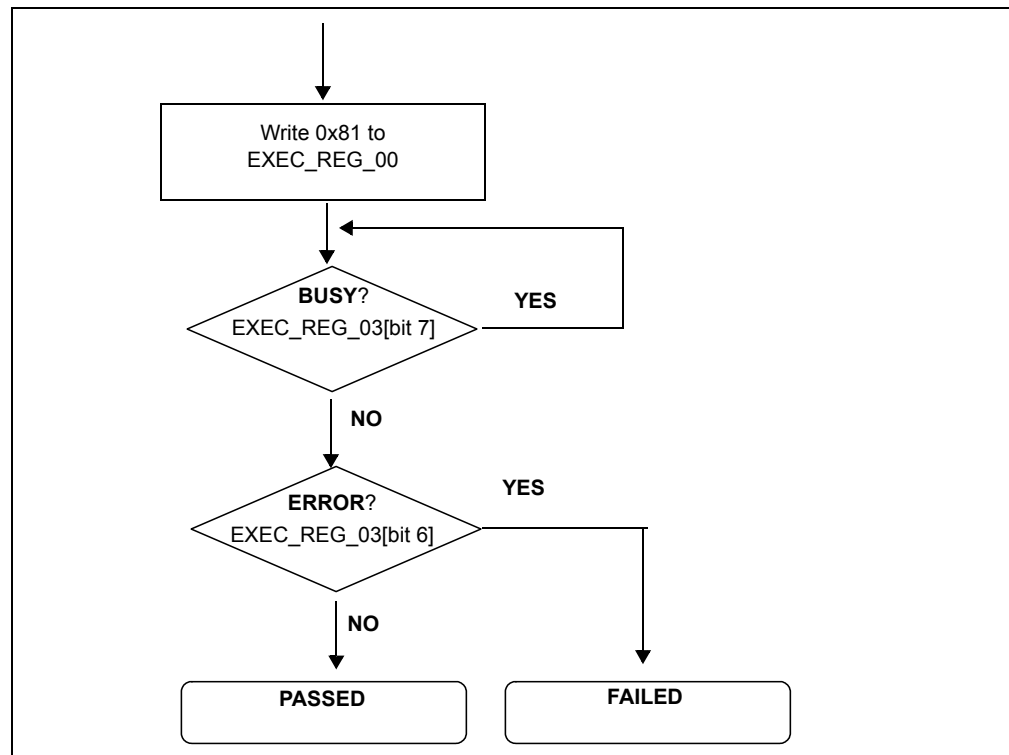
FIGURE B-6: PING-PONG LED PATTERN

DS1	DS2	DS3	DS3
On	Off	Off	Off
Off	On	Off	Off
Off	Off	On	Off
Off	Off	Off	On
Off	Off	On	Off
Off	On	Off	Off
On	Off	Off	Off

B.7.2 Timer1 Test

Timer1 Test provides a basic test of the auxiliary clock used for the RTC. The 28-Pin Demo Board employs a 32 kHz crystal on TIMER1 LP Oscillator to provide a 1-second interval for the RTC emulated device. The test begins by clearing the one-second flag, then waits a maximum of 1.1 seconds to see if the flag is reset. Alternatively, the HOST can test the RTC clock by examining EXEC REGISTER 0x02 at logically timed intervals.

FIGURE B-7: TIMER1 TEST



B.7.3 ADC Display

The ADC Display begins by displaying hex values 0x0A, 0x0D and 0x0C in sequence to signify “ADC” test.

After the opening display sequence, the LEDs displays the Most Significant 4 bits of the ADC result measuring channel AN0 (potentiometer RP1). As RP1 is manually turned from one extreme to the other, the LED display should range from binary 0000 to 1111. The firmware must be forced from this mode by command or Reset.

B.7.4 RTC Display

The RTC Display displays the Least Significant 4 bits of EXEC_REG_02 (RTC SECONDS). The LEDs should be seen to increment once per second in a binary progression (0x0-0xF) three times, then (0x0-0xB) once as the RTC SECONDS value increments from 0x00 to 0x3B (59 decimal). The firmware must be forced from this mode by command or Reset.

B.7.5 Communications Connector Display

This feature displays the state of the 4 signal pins on the communications connector.

LED1: PIN 1 (AUX1)

LED2: PIN 4 (SDA)

LED3: PIN 5 (SCL)

LED4: PIN 6 (AUX2)

Normally, LED2 and LED3 should be illuminated if the I²C bus has active pull-ups and the SCK is not held low. The firmware must be forced from this mode by command or Reset.

B.7.6 RESET Display

The RESET Display is executed on Reset/power-up or by command. The sequence is a series of individual displays as follows:

1. PING-PONG display: cycles: 2 (i.e., LED1,2,3,4,3,2,1,2,3,4,3,2,1)
2. Blink all LEDs in unison: cycles: 1 (i.e., 1 sec OFF, 1 sec ON, 1 sec OFF)
3. ADC display (described above): cycles: perpetual

B.7.7 1 Second Blink

Blink all LEDs in unison at 1-second intervals (i.e., 1 sec OFF, 1 sec ON, 1 sec OFF),... This feature uses time based on the Timer1 low-power oscillator and external 32 kHz tuning fork crystal.

PICKit™ Serial Analyzer User's Guide



WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
<http://support.microchip.com>
Web Address:
www.microchip.com

Atlanta

Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston

Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago

Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas

Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit

Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo

Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles

Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara

Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto

Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office

Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney

Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing

Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu

Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Fuzhou

Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

China - Hong Kong SAR

Tel: 852-2401-1200
Fax: 852-2401-3431

China - Qingdao

Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai

Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang

Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen

Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Shunde

Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

China - Wuhan

Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian

Tel: 86-29-8833-7250
Fax: 86-29-8833-7256

ASIA/PACIFIC

India - Bangalore

Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

India - New Delhi

Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune

Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama

Tel: 81-45-471-6166
Fax: 81-45-471-6122

Korea - Gumi

Tel: 82-54-473-4301
Fax: 82-54-473-4302

Korea - Seoul

Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Penang

Tel: 60-4-646-8870
Fax: 60-4-646-5086

Philippines - Manila

Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore

Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu

Tel: 886-3-572-9526
Fax: 886-3-572-6459

Taiwan - Kaohsiung

Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei

Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok

Tel: 66-2-694-1351
Fax: 66-2-694-1350

12/08/06

EUROPE

Austria - Wels

Tel: 43-7242-2244-39

Fax: 43-7242-2244-393

Denmark - Copenhagen

Tel: 45-4450-2828

Fax: 45-4485-2829

France - Paris

Tel: 33-1-69-53-63-20

Fax: 33-1-69-30-90-79

Germany - Munich

Tel: 49-89-627-144-0

Fax: 49-89-627-144-44

Italy - Milan

Tel: 39-0331-742611

Fax: 39-0331-466781

Netherlands - Drunen

Tel: 31-416-690399

Fax: 31-416-690340

Spain - Madrid

Tel: 34-91-708-08-90

Fax: 34-91-708-08-91

UK - Wokingham

Tel: 44-118-921-5869

Fax: 44-118-921-5820