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.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, dsPIC® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.
# Table of Contents

## Chapter 1. Introduction
1.1 Introduction ................................................................................. 1  
1.2 Highlights ................................................................................... 1  
1.3 What is the MCP2510 Development Kit ...................................... 1  
1.4 Multiple Uses for the MCP2510 Development Kit ....................... 2  
1.5 How the MCP2510 Development Kit Helps You ....................... 3  
1.6 Host Computer System Requirements ........................................ 3  

## Chapter 2. Installation
2.1 Introduction .................................................................................. 5  
2.2 Highlights ................................................................................... 5  
2.3 MCP2510 Development Kit Components .................................... 6  
2.4 Installing the Hardware ................................................................. 7  
2.5 Installing the Software ................................................................. 7  

## Chapter 3. Getting Started
3.1 Introduction .................................................................................. 9  
3.2 Highlights ................................................................................... 9  
3.3 Software Overview ..................................................................... 9  
3.4 Starting the Program .................................................................. 12  
3.5 Selecting the LPT Port ................................................................. 12  
3.6 Hardware Overview ................................................................... 13  

## Chapter 4. The Software Templates
4.1 Introduction .................................................................................. 15  
4.2 Highlights ................................................................................... 15  
4.3 MCP2510 Register View Template ............................................. 15  
4.4 Basic Template ........................................................................... 21  
4.5 Menus ......................................................................................... 27
Chapter 1. Introduction

1.1 INTRODUCTION

This user's guide is written assuming some basic knowledge of the CAN protocol and terms defined by CAN. Those users who do not have this basic CAN knowledge are encouraged to read Microchip's “CAN Basics” application note (AN713), the MCP2510 data sheet (DS21291) and/or the Bosch GmbH CAN specification, version 2.0. AN713 and the MCP2510 data sheet are available on Microchip's web site at www.microchip.com.

1.2 HIGHLIGHTS

This chapter discusses:

• What is the MCP2510 Development Kit
• Multiple Uses for the MCP2510 Development Kit
• How the MCP2510 Development Kit Helps You
• Host Computer System Requirements

1.3 WHAT IS THE MCP2510 DEVELOPMENT KIT

The MCP2510 Development Kit is a two-node Controller Area Network (CAN) tool that can be used in the evaluation/implementation of the MCP2510 stand-alone CAN controller. The software allows manipulation of the MCP2510 at the bit and byte levels with one template, while providing high-level control with a second template.

• One node is controlled by the PC that acts as a microcontroller using the provided software. This node can be used for basic MCP2510 evaluation/development and will be referred to as node0 throughout this document.

• The second node is controlled by a microcontroller that is programmed by the user as part of device validation and/or system development.

The two nodes are connected via a CAN bus that is also routed off-board through a connector, allowing the target board to be connected to an external CAN bus. This node will be referred to as node1 throughout this document.
1.4 MULTIPLE USES FOR THE MCP2510 DEVELOPMENT KIT

The MCP2510 Development Kit is used for evaluation of CAN node development utilizing the MCP2510. The tool is ideal for beginner CAN designers and/or those new to the MCP2510. Basic input and output functionality can be easily demonstrated by transmitting and receiving CAN messages.

The MCP2510 Development Kit also has multiple board configurations (via cutting traces between jumper holes) that allow the user to customize as needed.

1.4.1 Use as an Evaluation Tool

Evaluation of the CAN protocol and the MCP2510 is easily accomplished by utilizing the software supplied with the kit. The software controls only one of the two nodes on the board. There are two templates that can be used, depending on what end result is desired. See Chapter 3, “Getting Started”, for more details on the templates.

1. The Register template is used primarily for configuring/controlling individual registers in the MCP2510. All of the registers are modifiable, both bit-by-bit and byte-by-byte. This template allows the user to manipulate the MCP2510 registers and observe the effect on functionality. Different configurations can be set up and tested. Communication on the CAN bus is, typically, of secondary importance in this template.

2. The Basic template is designed to observe the MCP2510 while on a CAN bus. Direct register manipulation is not possible with this template, though bit rates and messages can be changed. Timed transmissions can be achieved using this template. This template also serves as a simple bus monitor.

1.4.2 Use as a Development Tool

The MCP2510 Developer's Tool can assist in development of a MCP2510-based CAN node by utilizing node 1, either by itself, or in conjunction with node 0. Example, node0 could be used as a simple bus monitor (using the basic template) in order to monitor the development of node1 (microcontroller node).

Another use of node0 may be used to set/verify bit timings, masks and filters.
1.4.3 Multiple Configurations Possible

There are multiple board configurations which give the user versatility in node development/evaluation.

• The board has three oscillator sockets allowing multiple oscillator configurations.
• The board has multiple CAN bus configurations (see Chapter 5, "Reconfigure the Hardware").
• Prototyping areas are available to allow the use of other MCUs that are not supported by the provided sockets and to change the CAN’s physical layer.

Some of the configurations are achieved by trace cutting and installing jumpers. The board configurations and jumper locations are discussed in detail in Chapters 5 and 6.

1.5 HOW THE MCP2510 DEVELOPMENT KIT HELPS YOU

The MCP2510 Development Kit can help module and system designers get up to speed with the MCP2510 and aids in the development of the MCP2510 into a CAN bus.

By using this tool, the time to design completion can be greatly reduced. Furthermore, the multiple configurations makes development versatile and efficient.

1.6 HOST COMPUTER SYSTEM REQUIREMENTS

The software will run on Windows® 95/98 or Windows NT®. Windows NT requires drivers (port95nt.exe) that are downloadable from www.kvaser.se or by searching the internet for "port95nt".
Chapter 2. Installation

2.1 INTRODUCTION

This chapter describes the procedure for installing the MCP2510 Development Kit.

2.2 HIGHLIGHTS

The items discussed in this chapter are:

- MCP2510 Development Kit Components
- Installing the Hardware
- Installing the Software
2.3 MCP2510 DEVELOPMENT KIT COMPONENTS

The MCP2510 Development Kit consists of these items:

• MCP2510 Development Board (contains PICmicro® MCU with demo firmware)
• Male-to-male DB25 cable
• CD-ROM with program software, User’s Guide and other supporting documents and code samples (forthcoming)
• Power cable and adapter
• Microchip Technical Library CD-ROM

FIGURE 2-1: MCP2510 Development Kit Components
2.4 INSTALLING THE HARDWARE

2.4.1 Required
The target board is connected to the PC via the provided 25-pin parallel cable (a standard male-to-male DB25 cable). LPT addresses supported are 0x3BC, 0x378 and 0x278.

Power to the target board is supplied by the included 9V power adapter.

2.4.2 Optional
There are two DB9 connectors:

One is a RS-232 interface for connecting to the PC as applications permit. This connector is connected to node1 and is typically used during PICmicro® MCU development to assist the user to design and/or debug (e.g., it may be used to print register and/or receive buffer contents to the PC screen for debugging).

The other connector is a CAN bus interface that can be used to connect the board to an existing CAN bus. The pinout for this connector follows the defacto standard recommended pinout of pin 7 = CANH and pin 2 = CANL.

![DB9 CAN Bus Connector]

2.5 INSTALLING THE SOFTWARE

Insert the CD into the PC. Click the Start button and select Run. Enter the path to the file wc32n.exe or select Browse and find the file. Alternate method: Through Windows Explorer®, run the file named wc32n.exe. Follow the instructions for installing the software program.
3.1 INTRODUCTION

This chapter explains how to set up the MCP2510 Development Kit for basic evaluation operation.

3.2 HIGHLIGHTS

The items discussed in this chapter are:

- Software Overview
- Starting the Program
- Selecting the LPT Port
- Hardware Overview

3.3 SOFTWARE OVERVIEW

The software is an easy-to-use program with two templates that perform different functions. Chapter 4, “The Software Templates”, details the operation of the templates and menu functions.

There are a couple of symbols that need explanation:
- A ‘$’ in front of the numbers represent hexadecimal.
- An ‘x’ after the number indicates a CAN protocol extended message. To type in an extended message, put the ‘x’ before the numbers to indicate that an extended message is being entered.

3.3.1 MCP2510 Register View Template

This template allows low-level control of the MCP2510 and would typically be used to evaluate/test the MCP2510 at the bit level. All of the registers required for complete configuration are available in this template.
FIGURE 3-1: MCP2510 Register View Template

The following windows are available in this template:

- **Status**
  The status window shows the contents of CANSTAT register (operation mode and interrupt flag codes), TX and RX error counts and EFLG register contents.
  Additionally, there are buttons to clear the overflow flag bits in the EFLG register.

- **Message Filters**
  The masks and filters are configurable in this window, allowing the user to set up and test for message acceptance. The Message Filters window allows messages to be tested against the masks and filters without physically going on a bus.

- **Physical Layer**
  The three CNF registers used for all CAN bit timings are configured in this window.

- **Configuration**
  TXRTSCTRL, BFPCCTRL, CANINTF, CANINTE and CANCTRL are all modified from this window. These are the control and flag registers.

- **Transmit**
  The transmit window controls the buffer contents for the transmit registers including TXBnCTRL, the identifier registers and the data registers.

- **Receive**
  This window contains all of the buffer contents for the receive buffers including RXBnCTRL, the identifier registers and the data registers.
3.3.2 Basic Template

The Basic Template is a high-level tool that focuses on CAN bus traffic. This template would typically be used to observe the MCP2510 while on the bus. Node1 development can be evaluated by using the basic template with node0.

**FIGURE 3-2: Basic Template**

The following windows are associated with the Basic Template:

- **Bus Status**
  - This window, labeled MCP2510 CAN Controller, provides several pieces of information about the status of the bus, including nominal bus loading, status of the node (on or off the bus) and bus bit rate.

- **Output**
  - The output window displays the messages that are received and transmitted. A time stamp indicates either delta times or running times between messages.

- **History List**
  - The history list window is used to collect transmitted messages for saving to a file. This file can be opened later and messages can be selected for retransmission. Some or all messages can be selected for transmission. This window works in conjunction with the Timed Transmissions window.

- **Timed Transmissions**
  - The Timed Transmissions window is used to send the messages in the History List window. The messages can be sent either one time (one shot) or repeated at regular intervals (cyclic).

- **Message Format Window**
  - This window determines the format of the displayed data in the output window. The default is Standard Text Format, which displays the message data as normal data.
3.4 STARTING THE PROGRAM

This section discusses the steps required to start the program.

1. To run the software, either select Start > Programs > Microchip > CANKing or find WC32.exe using Windows Explorer.

2. A dialog box will appear stating that putting this system on a CAN bus without configuring properly may adversely affect the bus. Click Okay.

3. A window will appear prompting to open a template or a project. If this is the first time running the program, select Template (no projects exist yet).

4. Select a template to open (Basic or MCP2510 Register View).

3.5 SELECTING THE LPT PORT

When starting the program for the first time after installation, it may be necessary to select the proper LPT port address from the available list (Options > MCP2510...).
3.6 HARDWARE OVERVIEW

The target board consists of two CAN nodes (MCU, MCP2510) a transceiver, an embedded CAN bus and support components.

The two nodes are connected to the embedded bus. By default, the embedded CAN bus is connected to the CAN connector (DB9), which is a link to an external CAN bus.

The support components are defined as all of the components that interface with the nodes as controls, indicators and other peripherals.

Figure 3-3 shows the main components of the board.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPT Port</td>
<td>The link between the MCP2510 and the PC that acts as the MCU for node0. The parallel port is used to allow the PC to communicate with MCP2510 via SPI™.</td>
</tr>
<tr>
<td>COM Port</td>
<td>The communications port (COM) is connected to the PICmicro MCU sockets (USART pins) via a MAX-232 device so that serial communication is possible between the PICmicro MCU and PC.</td>
</tr>
<tr>
<td>Oscillators</td>
<td>The three oscillator socket's outputs are connected together by default, so only one oscillator is needed to clock both MCP2510s and the PICmicro MCU. By cutting traces and installing jumpers, other oscillator configurations can be achieved. See Chapter 5, “Reconfigure the Hardware”, for more detail on configuring the oscillator sockets.</td>
</tr>
<tr>
<td>PICmicro MCU Sockets</td>
<td>Three sockets are provided to give the user a wide range of PICmicro MCUs to choose from when developing firmware.</td>
</tr>
<tr>
<td>MCU Prototype Area</td>
<td>This area was created for prototyping MCUs that are not supported with the sockets or for prototyping complete CAN nodes.</td>
</tr>
<tr>
<td>Non-Volatile Memory</td>
<td>Use of the 64-kbit SPI™ EEPROM is defined by the user. It is on the same SPI bus as the MCP2510. Therefore, care has to be taken to utilize the chip selects properly.</td>
</tr>
<tr>
<td>MCP2510</td>
<td>The MCP2510 is the interface between the CAN bus and the MCU.</td>
</tr>
<tr>
<td>CAN Transceiver</td>
<td>The CAN transceiver converts the differential signal on the bus to digital levels for the CAN controller and vice versa.</td>
</tr>
<tr>
<td>LED Banks</td>
<td>The LED banks reflect the state of many of the pins on the MCP2510.</td>
</tr>
<tr>
<td>RTS Buttons</td>
<td>These buttons are used to request transmission of the corresponding MCP2510’s transmit buffer if the pin is configured as RTS inputs or used as digital inputs.</td>
</tr>
<tr>
<td>CAN Connector</td>
<td>The CAN connector is used to connect the MCP2510 Development Kit to an external bus.</td>
</tr>
</tbody>
</table>
FIGURE 3-3: MCP2510 CAN Development Board
Chapter 4. The Software Templates

4.1 INTRODUCTION

There are two templates included with the MCP2510 Development Kit that give the user low-level bit control or high-level message control of the MCP2510.

4.2 HIGHLIGHTS

The items discussed in this chapter are:
• MCP2510 Register View Template
• Basic Template
• Menus

4.3 MCP2510 REGISTER VIEW TEMPLATE

This template allows low-level control of the MCP2510. This template is typically used to evaluate/test the MCP2510 at the bit level. All of the registers required for complete configuration are available in this template.

Register values can be changed on both a byte level and a bit level. To modify the register on a byte level, simply enter the value in the boxes next to the register names. Notice that the bit values will reflect the entered byte values.

To modify the registers at the bit level, double-click the desired bit. The bit will toggle for each double-click and the byte representation will be reflected next to the register name.

The bit boxes are only modifiable when unshaded. Shaded bit boxes are read-only bits.
This section discusses each of the template windows in detail.

### 4.3.1 Status Window

The Status Window displays the contents of the CANSTAT register, the EFLG register and the counts for the receive and transmit error counters. In addition, it contains buttons to clear each of the receive buffers’ overrun conditions.

The condition of the registers are also shown (e.g., $\text{CANSTAT} = 80\text{h}$ displays the condition as Configuration mode with no interrupts pending).
4.3.2 Message Filters Window

This window is used to set up and test mask and filter combinations with different message identifiers. Each mask and filter can be tested without actually writing the configuration to the MCP2510. This is done to allow the user to test different configurations against message identifiers that would appear on the bus.

For example, Figure 4-3 shows an identifier of $155\text{h}$ matching up against filter RXF1 and shows that the message would be accepted into receive buffer 0.

When the desired mask and filter combinations are achieved, the values can be written to the MCP2510 by clicking the Write button.

**Note:** The masks and filters can be written only when the MCP2510 is in Configuration mode.

![Message Filter Window](image)

**FIGURE 4-3:** Message Filter Window
4.3.3 Physical Layer Window

The Physical Layer window is where the CAN bus rate is configured. The user has bit-level control of the three CNF registers (CNF1, CNF2, and CNF3) that set up all items required for CAN bit timing, including the time quanta (TQ), bit segments, the synchronization jump width (SJW) and the baud rate prescaler (BRP).

The calculated bit rate is shown at the bottom of the window. For this calculation to be correct, the oscillator value must be correct. To change oscillator values, select Options > MCP2510… from the menu bar.

**Note:** The CNF registers can be modified only when the MCP2510 is in Configuration mode and will display shaded in all other modes of operation.

![MCP2510 Physical Layer](image)

**FIGURE 4-4:** Physical Layer Window
4.3.4 Configuration Window

TXRTSCTRL, BFPCTRL, CANINTF, CANINTE and CANCTRL are all modified from this window. These are the control and flag registers for the MCP2510.

4.3.4.1 TXRTSCTRL

This register configures the RTS pins (TXnRTS) as either request-to-send or as digital inputs. The pin values are displayed in the register if configured as digital inputs.

4.3.4.2 BFPCTRL

This register configures the receive buffer full pins (RXnBF) as buffer full interrupts or digital outputs. The value of the pins are displayed if configured as digital outputs.

4.3.4.3 CANINTF

CANINTF is the flag register for the eight sources of interrupts.

4.3.4.4 CANINTE

This register is the interrupt enables for the eight interrupt sources. Enabled interrupts are mapped to the INT pin.

4.3.4.5 CANCTRL

CANCTRL sets the modes of operation and the clock out enable and prescaler (CLKOUT pin).

![MCP2510 Configuration Window](image)

**FIGURE 4-5:** Configuration Window
4.3.5 Transmit Window

The Transmit window controls the buffer contents for the three transmit registers, including TXBnCTRL, the identifier registers and the data registers. The transmit buffers are selected using the Tx Buffer pull-down box.

Like the other register windows, the Transmit Window maps the byte values to the bit boxes. Entering data into the CAN ID box maps to multiple registers (SIDH, SIDL, EID8 and EIDO). Example: Entering 1FFFFFFFh in the CAN ID box maps all '1s' to SIDH, SIDL, EID8 and EIDO.

**FIGURE 4-6: Transmit Window**
4.3.6 Receive Window

This window contains all of the buffer contents for the receive buffers, including RXBCTRL, the identifier registers and the data registers. RXB0CTRL and RXB1CTRL are the only registers in this window that are not read-only, as indicated by the unshaded bit locations. These two registers set up the receive modes and enables/disables the rollover function.

![Receive Window Diagram]

**FIGURE 4-7: Receive Window**

**Note:** Selecting *Messages > MCP2510 Eval Board > Receive Buffer* (or Transmit Buffer) while holding down the shift key will open up duplicate windows so multiple transmit or receive windows can be monitored simultaneously.

4.4 BASIC TEMPLATE

The Basic Template is a high-level tool that focuses on CAN bus traffic. The user only has high-level control of the MCP2510 (i.e., no direct register control) that includes:

- Configuring the bus rate
- Changing modes of operation (Configuration and Normal)
- Configuring a transmit register (the register number is predefined)
- Transmitting messages
- Resetting the MCP2510
Other functions of the basic template are receiving messages, saving transmitted messages, starting timed transmissions and observing the bus loading. Details of these functions can be found later in this section under the details of the individual windows.

**Note:** Node0 is configured to receive ALL messages by default while in the basic template. The user has the option to maintain MCP2510 configurations when switching templates by selecting *Options > MCP2510...* and deselecting the Reset MCP2510 on Opening box. Keep in mind that deselecting this box may create unforeseen problems if the MCP2510 was not configured properly previously.

This template would typically be used as a simple bus monitor that could be used to evaluate the MCP2510 on a CAN bus or assist in development by monitoring how node1 is operating.

![FIGURE 4-8: The Basic Template](image-url)
4.4.1 Bus Status Window

This window, labeled MCP2510 CAN Controller, provides several pieces of information about the status of the bus, including nominal bus loading, status of node (on or off the bus) and bus bit rate. The tabs at the top of the window toggle between the Bus Statistics and Bus Parameters view.

In the Bus Statistics view, the bus loading, bus parameters and bus status may be viewed. The bus loading shows the nominal load on the CAN bus as a percentage and as a number, as well as the total number of messages sent and received. The bus parameters simply reflect the parameters that were set in the Bus Parameters view (i.e., bit rate, number of TQ, bit segment lengths and the location of the sample point). The bus status shows the mode of operation, error states and gives the user the ability to switch modes of operation between Normal and Configuration.

The Bus Parameters view allows configuration of the bus rate, the sample point, the synchronizing jump width (SJW) and switching between Normal and Listen-Only modes of operation.

![Bus Status Window Diagram]

*FIGURE 4-9: Bus Status*
4.4.2 Output Window

The output window displays the messages that are received and transmitted by node0. This window can be reconfigured to display messages in different formats. The formats are changed using a combination of the Message Format window and the right mouse button.

Changing the format type and/or the properties in the Message Format window alters the display in the output window. The numeric base (base 8, base 10 or base 16) can be changed. The time stamping function can also be changed to either a running time or a delta time. See Section 4.4.5, ‘Message Formats Window’, for more details.

Clicking the right mouse button while in this window brings up a menu that includes auto-scroll, fixed positions and cut/paste functions.

Auto-Scroll appends new messages to the bottom of the displayed messages so all messages are captured in the window. This feature allows the user to observe the order and frequency of messages on the bus.

Selecting Fixed Positions effectively assigns a slot for each message identifier. As messages with the same ID are received or transmitted, they overwrite the data contents of the previous message with the same ID. Used in conjunction with the delta time feature, the frequency of each message type can be observed.

The cut and paste menu items are standard operating system features.

FIGURE 4-10: Output Window
4.4.3 History List Window

The History List window is used to collect transmitted messages for saving to a file. This window is not opened by default when opening a new template and is opened by selecting View > History List. Once opened, every transmitted message is captured in the History List window for saving to a file, if desired.

One or more messages can be selected for retransmission. There is also a button for sending all the captured messages.

**Note:** The History List window works in conjunction with the Timed Transmission window. Messages can be retransmitted once or continually at defined intervals. See the Timed Transmission description for more details.

**FIGURE 4-11: History List Window**
4.4.4 Timed Transmissions Window

The Timed Transmissions window is used to send the messages in the History List window, either one time or continually at timed intervals.

The One Shot mode sends the selected (in the History List window) messages once each time the send buttons are pressed in the History List window.

The Cyclic selection sends the selected messages at regular selectable intervals (100 ms – 10 s). The other selections, Just-in-Time and Manual are not available in this software.

![Timed Transmission Window](image1)

**FIGURE 4-12:** Timed Transmission Window

4.4.5 Message Formats Window

This window determines the format of the displayed data in the output window. The default is Standard Text Format, which displays the message data as normal data. The properties of the Standard Text format can be changed by pressing the **Properties** button while highlighted. The properties are the numeric base and whether or not to display the message time stamp as a running total or as a delta.

![Message Formats Window](image2)

**FIGURE 4-13:** Message Formats Window
4.5 MENUS

The menu items, for the most part, are identical in both the register and basic template views. The few exceptions will be indicated in the descriptions below. Only two menu items will be discussed in this section, as the others are self-explanatory.

4.5.1 Messages menu

In the Basic Template, this menu contains the message types to be transmitted. Selecting message types opens a window that the user can configure to transmit a message. Example, selecting Messages > Universal > Universal opens a window for transmitting a normal (up to eight bytes) message.

The CAN Kingdom message type menu items are for transmitting messages that adhere to the CAN Kingdom higher-layer protocol (HLP).

4.5.2 Options menu

The options menu contains the configurable properties of both the hardware and the software.

The default numeric base is selected by Options > Global. Auto-save switches are also contained here.

Options > MCP2510... is where the LPT port address is selected. It is also where the oscillator frequency is set in software. Additionally, this menu contains a switch that determines whether or not the MCP2510 will be reset on file open. This feature is for those users who want to switch between templates without changing the MCP2510 configuration (e.g., masks and filters are set to receive ALL messages in the Basic template, unless this box is deselected).

Note: The oscillator frequency must be set in software to match the hardware so the software can configure the bit timing registers correctly. The formulas for bit timing contain an oscillator frequency.

FIGURE 4-14: Menu Bar
Chapter 5. Reconfigure the Hardware

5.1 INTRODUCTION

The MCP2510 Development Kit has several different configurations that make it a versatile tool.

5.2 HIGHLIGHTS

The items discussed in this chapter are:
• Node Configurations
• Oscillator Routing
• The Jumper Settings

5.3 NODE CONFIGURATIONS

The versatility of the MCP2510 Development Kit is possible by changing the configuration of the board. Refer to Section 5.5 “The Jumper Settings”, for a list of the various jumper settings required to change board configurations. There are five common configurations.

Note: All jumper locations are shorted by default, unless otherwise specified. The traces must be cut if the installed jumpers are to function properly.

5.3.1 PC Node Non-Distributed (one node, no CAN bus)

This is a default configuration in which the PC acts as a microcontroller for node0. This configuration does not have to be connected to the CAN bus because no CAN communication occurs. This configuration is used for evaluation of, or familiarization with, the MCP2510. The Register Template would be used. The masks, filters and register functions can easily be evaluated in this configuration.
5.3.2 PC Node Distributed (one node, on CAN bus)

This configuration is the same as above, with the exception that it is connected to an external CAN bus via the CAN connector (DB9).

Typically, this configuration would be used to further evaluate the MCP2510 by observing how it functions on an external CAN bus. Experimentation with bit timings, masks and filters, interrupts, RTS pins, etc. can be performed while using the Register Template. While in the configuration, simple bus monitoring can be achieved using the Basic Template.

5.3.3 Two Node Embedded System

This configuration utilizes node0 and node1 to create a two node embedded system (no external bus).

This configuration can be used for evaluation or development. The microcontroller firmware is being developed at this point. As an example, the firmware may be written to observe how the MCP2510 uses masks and filters to accept/reject messages. SPI modules and interrupt handlers may be under development at this stage.

5.3.4 Two Node Distributed System

This configuration places both nodes on the CAN bus.

One scenario utilizes the microcontroller node as the node under development, while the PC node is simply a bus monitor to assist in debugging.

5.3.5 Microcontroller Distributed System (One Node)

This configuration places the microcontroller node (node1) on the CAN bus.

5.4 OSCILLATOR CONFIGURATIONS

There are three oscillator sockets. By default, both MCP2510s and the PICmicro MCU sockets use a common oscillator. Since all three socket outputs are tied together, the oscillator can be placed in any socket.

**Warning:** Care must be taken when installing more than one oscillator. The jumper settings must be correct or contention will occur at some or all of the device oscillator inputs. This may cause catastrophic results.

It is possible for each node and the microcontrollers to have their own oscillator by configuring the jumpers as described later in this chapter.
5.5 THE JUMPER SETTINGS

There are multiple jumper settings that allow the MCP2510 Development Kit to be configured in different ways.

There are jumpers that allow the transceivers to be disconnected so another physical layer can be added either in the prototyping area or by adding a daughter board that is manufactured by a third party. Contact Kvaser AB® for more information (www.kvaser.se).

There are also jumpers that disconnect the three oscillators from each other. A CAN bus terminating resistor (120 ohm) can be jumpered in. Finally, the LED banks can be disconnected, if so desired.

Note: All jumpers are shorted with a copper trace (bottom-side of board) by default and require cutting to open. Jumper pins can be soldered in the holes to allow jumper connectors to be used.

### TABLE 5-1: JUMPER DESCRIPTION

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP1</td>
<td>Connects LEDs for Node0</td>
</tr>
<tr>
<td>JP2</td>
<td>Connects Node0 MCP2551 RS pin to external bus (DB9 J4)†</td>
</tr>
<tr>
<td>JP3</td>
<td>Connects Node0 MCP2551 CANH pin to external bus (DB9 J4)†</td>
</tr>
<tr>
<td>JP4</td>
<td>Connects Node0 MCP2551 CANL pin to external bus (DB9 J4)†</td>
</tr>
<tr>
<td>JP5</td>
<td>Connects Node0 MCP2551 RXD pin to external bus (DB9 J4)†</td>
</tr>
<tr>
<td>JP6</td>
<td>Connects oscillator O2 to oscillator O1</td>
</tr>
<tr>
<td>JP7</td>
<td>Connects LEDs for Node1</td>
</tr>
<tr>
<td>JP8</td>
<td>Connects oscillator O3 to oscillator O1</td>
</tr>
<tr>
<td>JP9</td>
<td>Connects oscillator O3 to oscillator O2</td>
</tr>
<tr>
<td>JP10</td>
<td>Connects Node1 MCP2510 INT pin to PICmicro® RB0 pin</td>
</tr>
<tr>
<td>JP11</td>
<td>Connects Node1 MCP2551 RS pin to external bus (DB9 J4)†</td>
</tr>
<tr>
<td>JP12</td>
<td>Connects Node1 MCP2551 CANH pin to external bus (DB9 J4)†</td>
</tr>
<tr>
<td>JP13</td>
<td>Connects Node1 MCP2551 CANL pin to external bus (DB9 J4)†</td>
</tr>
<tr>
<td>JP14</td>
<td>Connects Node1 MCP2551 RXD pin to external bus (DB9 J4)†</td>
</tr>
<tr>
<td>JP15</td>
<td>Connects 120 ohm terminating resister to the CAN bus</td>
</tr>
</tbody>
</table>

† These jumpers are provided to disconnect the MCP2551 device from the bus so other physical layers can be used, including a third party daughter card available from Kvaser AB.
FIGURE 5-1: Jumper Locations
Appendix A. Schematics

A.1 INTRODUCTION

This section contains the schematics, which are also available on the included CD-ROM.
A.2 SCHEMATIC

Note: This schematic is available on the included CD-ROM (03-01522r1-S1.pdf).
Note: This schematic is available on the included CD-ROM (03-01522r1-S2.pdf).
Appendix B. FAQs on Configuring the MCP2510

B.1 INTRODUCTION

This section answers some frequently asked questions on configuring the MCP2510 to assist those who are new to the device.

B.2 HIGHLIGHTS

This section discusses:

• FAQs

B.3 FAQS

1. Why doesn’t the development tool successfully communicate on an external bus?

   There are several possible reasons why this is the case. The following are the most common problems:

   TABLE B-1: COMMON BUS COMMUNICATION PROBLEMS

<table>
<thead>
<tr>
<th>Condition</th>
<th>Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>The MCP2510 is off the bus (not in Normal mode).</td>
<td>Basic Template – Go on Bus button in the “Bus Status” window.</td>
</tr>
<tr>
<td>The LPT port is not configured correctly.</td>
<td>Make sure LPT address is correct (Options &gt; MCP2510…).</td>
</tr>
<tr>
<td>The bus rate is not set to match the bus.</td>
<td>Basic Template – Set the bus rate in the Bus Status window.</td>
</tr>
<tr>
<td></td>
<td>Register Template – change CNF registers.</td>
</tr>
<tr>
<td>Oscillator frequency not set to match hardware (Fosc is required in bit rate formula).</td>
<td>Set the oscillator frequency (Options &gt; MCP2510…).</td>
</tr>
<tr>
<td>Board is not connected to the external bus.</td>
<td>Connect to the external bus using the DB9 labeled CAN.</td>
</tr>
</tbody>
</table>

2. The node is acknowledging messages in the Basic template but not displaying them.

   The MCP2510 filters are not matching the incoming messages. This can occur because the Reset MCP2510 on Open is deselected, causing the old register contents (masks and filters) to remain unchanged when switching templates.

   Press reset, or select the Reset MCP2510 on Open box and reopen the template.
3. **I cannot enter values in the Physical Layer window.**
   The MCP2510 is not in Configuration mode. The three CNF registers are only modifiable while in Configuration mode. Change modes.

4. **The mask and filter registers do not change when pressing the write button.**
   The MCP2510 is not in Configuration mode. Masks and filters are changeable only in Configuration mode. Change mode.

5. **How do I enter extended IDs into a field?**
   Lead off the number with an 'x' which indicates extended frame (e.g., x12345).

6. **Are there daughter boards available that change the physical layer?**
   Yes. Contact Kvaser AB for details (www.kvaser.se).