

**MSK28335
&
MCK28335**

**TMS320F28335 DSC
Motion Starter Kits
& Motion Control Kits**



DSC Motion Solutions

User Manual

TECHNOSOFT
DSC Motion Solutions

MSWIN28335
MCWIN28335
User Manual

P091.046.UM.0911

Technosoft S.A.

Avenue des Alpes 20
CH-2000 NEUCHÂTEL
Switzerland

Tel: +(41) 32 732 5500

Fax: +(41) 32 732 5504

contact@technosoftmotion.com

<http://www.technosoftmotion.com>

Contents

Read this first
Overview and License

Chapter

1. Introduction
2. Installing the MSK28335 / MCK28335 kits
3. Motion Control Evaluation and Development Tools
4. Communication Module
5. Processor Evaluation Applications
6. Motion Control Graphical Analysis Tool (*MCK28335 kits*)
7. Brushless AC (BLAC), Brushless DC (BLDC) and Induction Motor Vector Control (IMVC) Motion Demos (*MCK28335 kits*)
8. DMC28x Developer demos

Appendices – Technical Specifications

- A. MSK28335 DSC board - Technical data
- B. Connecting external power modules to MSK28335 DSC board
- C. TMS320F28335 resources used by MSK28335 board
- D. PM50 V3.1 power module technical data
- E. MSK Level Adapter v1.0 board
- F. Motors Datasheets
 - escap[®] 22BC-8B brushless motor
 - Pittman 3441 series brushless motor
 - Technosoft MBE.300.E500 brushless motor
 - Technosoft XBR-55-06-602-CE brushless motor
 - Technosoft ASPMN000011 induction motor

Read This First

About this manual

Technosoft provides 4 development kits based on the Texas Instruments TMS320F28335 DSC (digital signal processor) motion controller:

- **MSK28335 Kit** - DSC Motion Starter kit (include DMC28x Developer - Lite)
- **MCK28335 Kit** – DSC Motion Control kit (include DMC28x Developer - Lite)
- **MSK28335 Pro Kit** – Professional version of the MSK28335 kit (include DMC28x Developer - Pro)
- **MCK28335 Pro Kit** – Professional version of the MCK28335 kit (include DMC28x Developer - Pro)

This book presents all the hardware and software components of these kits, except the development platforms: **DMC28x Developer - Lite** for MSK28335 and MCK28335 kits and **DMC28x Developer - Pro** for the professional versions. Please refer to the **DMC28x Developer - Pro** manual.

This book describes the **MSK28335 DSC board**, the **PM50** power module (*MCK28335 kits only*) and how to use them with the accompanying software:

- The DSC board / PC monitor programs for serial communication
- The TMS320F28335 DSC evaluation programs
- The graphical analysis tool (*MCK28335 kits only*)
- Brushless motor and induction motor motion demos (*MCK28335 kits only*)

How to Use This Manual

The goal of this book is to help you learn to use the hardware and software of the MSK28335 and MCK28335 kits. The book contains the following chapters:

- Chapter 1 describes the features and provides an overview of the MSK28335 and MCK28335 kits.
- Chapter 2 contains installation instructions, and lists the hardware and software tools you'll need in order to use the MSK28335 and MCK28335 kits
- Chapter 3 describes the structure of the kits software platform and lists the program components.
- Chapter 4 contains detailed information about using the communication monitor programs, both at MSK28335 DSC board level and at PC level.
- Chapter 5 contains detailed information about using the processor evaluation applications.
- Chapter 6 describes the graphical motion analysis tool, provided with the MCK28335 kits
- Chapter 7 describes the several motion demos included in the MCK28335 kits

Notational Conventions

This document uses the following conventions.

- Examples use **bold special typeface** for source code and normal special typeface for comments; interactive displays use **bold special typeface** for commands that you enter normal special typeface for the text the system displays (such as prompts, command output, error messages, etc.).

Here is a sample program listing:

```
MOV      AL,#0
LCR      #_SetSCIBaudrate
MOVW     DP,#_i
MOV      @_i,#0
MOV      AL,@_i
CMPB     AL,#7
```

Here is an example of a system prompt and a command that you might enter:

```
DSP>i<CR>
  Inspect data memory
  From [hexa]: 200<SP>
```

- In syntax descriptions, the instruction, command or directive is in a **bold** font and parameters are in *italics*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Syntax that is entered on a command is centered in a bounded box. Syntax that is used in a text file is left-justified in an unbounded box. Here is an example of a directive syntax:

```
.include filename
```

.include is the directive. This directive has one parameter, indicated by *filename*. When you use this directive, the parameter must be an actual filename.

- Square brackets ([and]) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don't enter the brackets themselves. Braces ({ and }) indicates a list. The symbol | (read as *or*) separates items within the list. Here's an example of an instruction that has optional parameters, some of them selected from an items list:

```
MONPC28X [ {COM1|COM2} ] [ {9600|19200|38400} ] [X] [P]
```

The **MONPC28X** command has several optional parameters. Two of them must be chosen (if used), from a list of possible items (COM1 *or* COM2 for example).

Information About Cautions and Warnings

This book may contain warnings.



CAUTION!

THIS IS AN EXAMPLE OF A WARNING STATEMENT.
A WARNING STATEMENT DESCRIBES A SITUATION THAT COULD
POTENTIALLY CAUSE HARM TO YOU, OR TO YOUR HARDWARE
SYSTEM

Related Documentation from Technosoft

DMC28x Developer Lite User Manual. (literature number UDMCDL.C) describes the DMC Developer Lite development platform for TMS320F28x DSP controllers.

DMC28x Developer Pro User Manual. (literature number UDMCDP.B) describes the DMC Developer Pro development platform for TMS320F28x DSP controllers.

Related Documentation from Texas Instruments

The following books describe the TMS320x2833x and related support tools. To obtain a copy of these documents, call the Texas Instruments Literature Response Center at (800) 477-8924. When ordering, please identify the book by its title and literature number.

TMS320C28x Assembly Language Tools User's Guide. (literature number SPRU513) describes the assembly language tools (assembler and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for the TMS320C28x device.

TMS320C28x DSP CPU and Instruction Set Reference Guide. (literature number SPRU430) describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x 16-bit fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

TMS320C28x Floating Point Unit and Instruction Set Reference Guide. (literature number SPRUE02) describes the floating-point unit and includes the instructions for the FPU.

TMS320C28x Optimizing C/C++ Compiler User's Guide. (literature number SPRU514) describes the TMS320C28x™ C/C++ compiler. This compiler accepts ANSI standard C/C++ source code and produces TMS320 DSP assembly language source code for the TMS320C28x device.

TMS320x2833x Analog-to-Digital Converter (ADC) Module. (literature number SPRU812) The TMS320C28x™ ADC module is a 12-bit pipelined analog-to-digital converter (ADC). The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample-and-hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits.

TMS320x2833x Boot ROM Reference Guide. (literature number SPRU963) The boot ROM is factory-programmable with boot-loading software. Boot-mode signals (general-purpose I/Os) are used to tell the bootloader software which mode to use. The Boot ROM also contains standard math tables such as SIN/COS for use in IQ math related algorithms.

TMS320x28xx, 28xxx DSP Peripheral Reference Guide. (literature number SPRU566) This overview guide includes all the peripherals available for TMS320x28xx / TMS320x28xxx devices.

-
- TMS320x28xx, 28xxx Enhanced Pulse Width Modulator (ePWM) Module Reference Guide** (literature number SPRU791) describes the main areas of the enhanced pulse width modulator that include digital motor control, switch mode power supply control, UPS (uninterruptible power supplies), and other forms of power conversion.
- TMS320x28xx, 28xxx High-Resolution Pulse Width Modulator (HRPWM)** (literature number SPRU924) describes the operation of the high-resolution extension to the pulse width modulator (HRPWM).
- TMS320x28xx, 28xxx Enhanced Capture (eCAP) Module Reference Guide** (literature number SPRU807) describes the enhanced capture module. It includes the module description and registers.
- TMS320x28xx, 28xxx Enhanced Quadrature Encoder Pulse (eQEP) Reference Guide** (literature number SPRU790) describes the eQEP module, which is used for interfacing with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine in high performance motion and position control systems. It includes the module description and registers.
- TMS320x28xx, 28xxx Enhanced Controller Area Network (eCAN) Reference Guide.** (literature number SPRU074) The enhanced 'Controller Area Network' (eCAN) module implemented in the 28x™ DSC is compatible with the CAN 2.0B standard (active). It uses established protocol to communicate serially with other controllers in electrically noisy environments.
- TMS320x28xx, 28xxx Serial Communication Interface (SCI) Reference Guide** (literature number SPRU051) describes the SCI, which is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format.
- TMS320x28xx, 28xxx Serial Peripheral Interface (SPI) Reference Guide** (literature number SPRU059) describes the SPI - a high-speed synchronous serial input/output (I/O) port - that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit-transfer rate.
- TMS320x28xx, 28xxx Inter-Integrated Circuit (I2C) Reference Guide** (literature number SPRU721) describes the features and operation of the inter-integrated circuit (I2C) module that is available on the TMS320x28xx and TMS320x28xxx digital signal processor (DSP).
- TMS320x2833x DSC External Interface (XINTF) Reference Guide.** (literature number SPRU949) The external interface (XINTF) on the 2812 device and the 2833x devices is an asynchronous bus that is used to interface to external devices and memory.
- TMS320F2833x Multichannel Buffered Serial Port (McBSP) Reference Guide.** (literature number SPRUFB7) The McBSP is used to connect to E1/T1 lines, phone-quality codecs for modem applications or high-quality stereo-quality Audio DAC devices. The McBSP receive and transmit registers are supported by a 16-level FIFO. This significantly reduces the overhead for servicing this peripheral.
- TMS320x2833x System Control and Interrupts Reference Guide.** (literature number SPRUFB0) This document contains descriptions of: memory (including Flash and OTP), code security module (CSM), clocking and low-power modes, 32-bit CPU-Timers, watchdog timer, general-purpose inputs/outputs (GPIO), peripheral frames, peripheral interrupt expansion (PIE) and external interrupts.

If you Need Assistance ...

If you want to ...	
Visit Technosoft online	World Wide Web: http://www.technosoftmotion.com/
Register your kits	World Wide Web: http://www.technosoftmotion.com/
Receive general information or assistance (see Note 1)	Email: contact@technosoftmotion.com
Ask questions about product operation or report suspected problems (see Note 1)	Fax: (41) 32 732 55 04 Email: hotline@technosoftmotion.com
Make suggestions about or report errors in documentation (see Note 1)	Mail: Technosoft SA Buchaux 38 CH-2022 Bevaix, NE Switzerland

Notes:

- 1) You need first to register your MSK28335 or MCK28335 kit in order to get free assistance and support.
- 2) Remember your Serial No / License No since you will need it to access User Pages from the web site.
- 3) For information about Texas Instruments products, use the World Wide Web address <http://www.ti.com>

FCC Warning

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Trademarks

MS-DOS and Windows are registered trademarks of Microsoft Corporation.

PC is trademark of International Business Machines Corporation.

Single License Agreement for End-User of the Software of the MSK28335 and MCK28335 kits

Important : To read with care

This is a legal agreement (the Agreement) between you, being the purchaser (either as private individual or as legal entity) and Technosoft for the software (SOFTWARE) of Technosoft included in the MSK28335 kit, MSK28335 kit Pro, MCK28335 kit and MCK28335 kit Pro which includes computer programs, corresponding disks, written documentation, and which can include an "online" or electronic documentation. If you install, copy or use in any way the SOFTWARE, you are agreeing to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, please return the unused SOFTWARE to Technosoft for a full refund.

SOFTWARE License

The SOFTWARE is protected by copyright laws and international treaty provisions and all other applicable laws on intellectual property. The SOFTWARE is not sold but granted with a license.

1. License Concessions

- ☞ **Grant of license.** Technosoft grants you the right to use one copy of the SOFTWARE or, instead, of any previous version, for the same system and on a single computer.
- ☞ **Storage / Network Server.** You also have the right to install and archive one copy of the SOFTWARE on a storage device such as a network server used solely to install, or to use the SOFTWARE on different computers linked to an internal network. However, you must obtain and dedicate a license for each individual computer on which the SOFTWARE is installed or runs from a storage device.

2. Other rights and limitations

- ☞ **No reverse engineering, decompilation and disassembly.** You may not reverse engineer, decompile or disassemble the SOFTWARE.
- ☞ **Dissociation of components.** The SOFTWARE is provided with a license as one single product. You may not disunite the modules in order to use them on more than one computer.
- ☞ **Lease.** You may not lend, rent or lease the SOFTWARE to a third party.
- ☞ **Transfer of SOFTWARE.** You may transfer your rights under this license Agreement on a permanent basis provided you do not retain any copy, you transfer the whole SOFTWARE (including all components and disks) all written materials and updates, and the recipient agrees to the terms of this Agreement. If the SOFTWARE is an update, any transfer must include all its prior versions.
- ☞ **Termination.** Without prejudice to all other rights, Technosoft may terminate the Agreement if you do not respect its terms. In that case, all copies of the SOFTWARE together with all its components should be destroyed.

3. Updates

If the SOFTWARE is an update of another product, you may use or transfer it only in conjunction with the updated product, unless you only erase the updated product. If the SOFTWARE is an update of a Technosoft product, you may use this updated product solely in conformity with the terms of the Agreement. If the SOFTWARE is an update of a component out of a package for which you obtained a one package license, then the SOFTWARE may be used and transferred solely as part of this software package but may not be separated for use on more than one computer.

4. Copyrights

All rights of ownership and copyrights relative to the SOFTWARE, to the accompanying written material and to each copy of the SOFTWARE are held by Technosoft or its suppliers. The SOFTWARE is protected by copyright laws and international treaties on intellectual property. Accordingly, you must treat the SOFTWARE as any other element protected by copyrights but you are authorized to (a) either make one copy of the SOFTWARE solely for backup or archival purposes, (b) or to transfer the SOFTWARE to a single computer, provided you keep the original solely for backup or archival purposes. You may not copy the Product manual(s) or written materials accompanying the SOFTWARE.

5. Warranties

☞ **Limited warranty.** Technosoft warrants that the SOFTWARE will perform substantially with specifications defined on the accompanying product manual(s) for a period of 90 days from the date of receipt. Any implied warranties on the SOFTWARE and hardware are limited to 90 days and one (1) year, respectively.

☞ **Customer remedies.** Technosoft 's entire liability and the customer's exclusive remedy shall be, at Technosoft 's option, either the return of the price paid or the replacement of the SOFTWARE that does not meet Technosoft 's Limited Warranty and which is returned to Technosoft with a copy of the receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or for a minimum of 30 days, which ever is longer.

☞ **No other warranties.** To the maximum extent permitted by applicable law, Technosoft disclaims all other warranties, either express, implied or legal, including but not limited to legal warranties of merchantability and fitness for a particular purpose, with respect to the SOFTWARE, the accompanying product manual(s) and written materials.

☞ **No liability for consequential damages.** Results obtained when using the SOFTWARE are not a guarantee for a good functioning of the components within the application. Such results are only indicative and do not release the user from testing under practical conditions.

To the maximum extent permitted by applicable law, Technosoft and its suppliers shall not be liable for any other damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or of any other information or other pecuniary loss) arising out of the use of or inability to use this SOFTWARE, even if Technosoft has been advised of the possibility of such damages. In any case, Technosoft's entire liability under any provision of this Agreement shall be limited to the amount actually paid for the SOFTWARE.

This Agreement is governed by the laws of Switzerland.

1 Introduction

This chapter provides an overview of the following development kits:

- **MSK28335 kit** - DSC Motion Starter kit
- **MCK28335 kit** – DSC Motion Control kit
- **MSK28335 Pro Kit** – Professional version of the MSK28335 kit
- **MCK28335 Pro Kit** – Professional version of the MCK28335 kit

These kits are based on the **MSK28335 DSC board** and allow you to experiment with and use the TMS320F28335 (**F28335**) DSC controller for digital motion control (**DMC**) applications.

The 'F28335 DSC controller is a 3.3V low voltage device, designed to implement advanced DMC applications, by integrating high performance DSC core and on-chip peripherals of a microcontroller into a single chip. The 'F28335 has all the peripherals needed to control two simultaneous 3-phase AC motors.

The **MSK28335 kit** is the basic kit. It is oriented towards the 'F28335 features evaluation but also offers the possibility to start developing DSC code using the assembler, linker and debugger included in the **DMC28x Developer - Lite** platform.

The **MCK28335 kit** also includes a 3-phase power module and a brushless motor, a graphical analysis tool for DMC applications and several ready-to-run motion demos. This kit provides a complete platform for motion applications evaluation and testing.

The **MSK28335 kit Pro** and the **MCK28335 kit Pro** are the professional versions of the above mentioned kits. These kits include the **DMC28x Developer - Pro** platform, a professional integrated development environment (IDE) including: assembly and C language debugger, trace functions, advanced graphical tools based on real time data acquisitions, reference generator, etc.

All the kits offer the possibility to control one or two electrical motors. The MSK28335 DSC board includes 2 connectors dedicated for interfacing with external power modules. Each connector provides all the necessary I/O signals to interface with a power module and drive an electrical motor.

Contents

1.1. Key Features

1.2. MSK28335 / MCK28335 Kits Hardware Overview

1.3. MSK28335 / MCK28335 Kits Software Overview

1.1 Key features

MSK28335 DSC board

- DSC motion controller TMS320F28335
- 256-kword 0-wait state SRAM external program and/or data memory
- 8-kword (16-kbytes) serial SPI-connected E2ROM
- 2 channels 12-bit D/A converter, simultaneous update, serial SPI-connected
- RS-232 and CAN field bus communication ports
- expansion connectors providing access to all the 'F28335 signals
- JTAG interface (XDS510/XDS510PP emulator connector)
- single power supply: $(5 \pm 10\%)V_{DC}$

MSK28335 kit

- **MSK28335 DSC board**
- **MSWIN28335** software platform including:
 - monitor programs
 - **PROCEV2833x** - processor evaluation applications
 - **DMC28x Developer - Lite** environment including assembler, linker and debugger
- **User Manuals:** MSK28335 / MCK28335 Kits, DMC28x Developer - Lite
- **TMS320F28x DSC Controllers** Reference Guide

MSK28335 kit Pro

- **MSK28335 kit with DMC28x Developer - Pro** replacing DMC28x Developer - Lite

PM50 power module board

- 3-phase PWM inverter: 36 V, 2.1 A, up to 25 kHz PWM frequency
- measurement of motor currents in all 3 phases and DC voltage
- incremental encoder port
- supply: 12-36V_{DC}

MCK28335 Motion Control Kit

- **MSK28335 DSC board**
- **PM50** power module board
- **3-phase brushless motor** equipped with 500-line quadrature incremental encoder and 3 Hall position sensors
- **MCWIN28335** software platform including:
 - monitor programs
 - **PROCEV2833x** - processor evaluation applications
 - IDE graphical analysis tool for digital motor control applications
 - two demos for brushless motor control: **BLDC** - brushless DC (trapezoidal) and **BLAC** - brushless AC (sinusoidal), one demo for **IMVC** – induction motor (vector control)
 - **DMC28x Developer - Lite** environment including assembler, linker and debugger
- **User Manuals:** MSK28335 / MCK28335 Kits, DMC28x Developer Lite
- **TMS320F28x DSC Controllers** Reference Guide

MCK28335 kit Pro

- **MCK28335 kit with DMC28x Developer - Pro** replacing DMC28x Developer - Lite

1.2 MSK28335 / MCK28335 Kits Hardware Overview

Figure 1.1 presents the block diagram of the **MSK28335 DSC board**. The board includes an RS-232 interface, used for communication with the PC and a CAN transceiver that permits to connect several MSK28335 DSC boards using a CAN-bus network.

The MSK28335 DSC board is equipped with 256-kword external SRAM memory that can be used as data memory, as program memory or both and an 8-kword serial SPI-connected E2ROM. For performance evaluation and debugging the MSK28335 DSC board includes a 2-channel 12-bit serial SPI-connected D/A converter with simultaneous update.

The JTAG connector offers compatibility with all programs using the XDS510 or XDS510PP emulator.

The MSK28335 DSC board offers direct access to all of the 'F28335 I/O signals through 4 expansion connectors. Two connectors contain the data bus, the address bus and the control signals required to add new interfaces. The other 2 connectors are dedicated for interfacing with power modules. Each of these connectors includes all the basic I/O signals required to control a DC, AC or step motor. Through these connectors the MSK28335 DSC board can be linked with one or two external power modules of various types and powers, adapted to specific applications. This approach is illustrated in the **MCK28335 Kits** where the MSK28335 DSC board is connected with PM50 power module using one of these 2 connectors. The 2nd connector may be used for example with another PM50 power module.

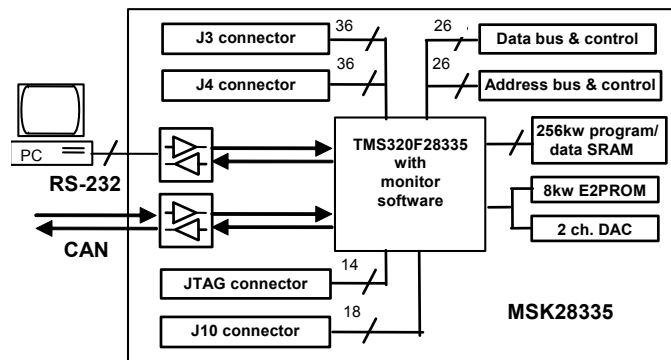


Figure 1.1. The block diagram of the **MSK28335** board

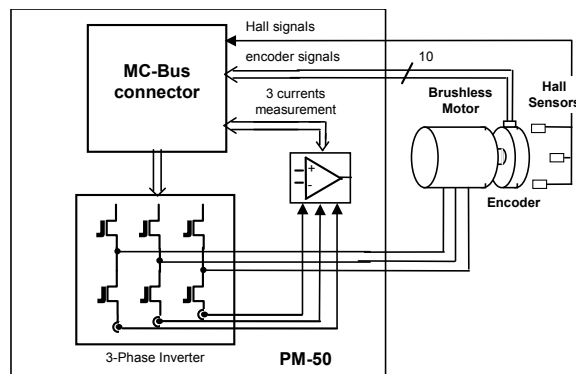


Figure 1.2. The block diagram of the **PM50** power module

Figure 1.2 presents the block diagram of the **PM50** board. Connectable with MSK28335 through J3 (PM50 V3.1), the PM50 is a 2.1A, 36V three-phase inverter, with connectors for motor phases, encoder and Hall signals.

1.3 MSK28335 / MCK28335 Kits Software Overview

The **MSK28335** and **MCK28335** kits contain several software applications, constituting a basic development platform, **MSWIN28335** in the case of the **MSK28335** kits, respectively **MCWIN28335** in the case of the **MCK28335** kits.

Both **MSWIN28335** and **MCWIN28335** platforms integrate a monitor communication program, as well as the 'F28335 DSC controller evaluation applications. Using the PC communication module, you can access the **MSK28335** board and execute any of these programs. You can integrate modules from these programs into your applications, and use the **MSK28335** platform to analyze and evaluate the results, as suggested in Figure 1.3.

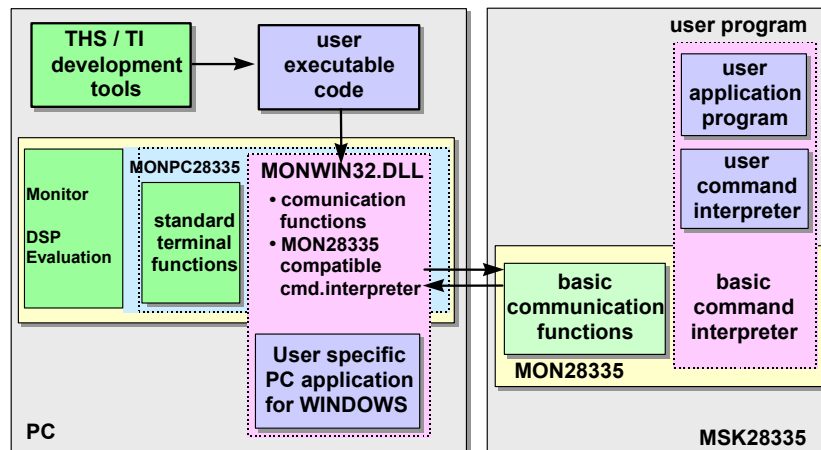


Figure 1.3. Basic communication programs structure

1.3.1 Basic Communication Software

The basic communication programs between the **MSK28335 DSC board** and the PC are **MON28335** (on the **MSK28335 DSC board**) and **MONPC28x Monitor** (on the PC). Figure 1.4 shows the basic structure of these programs.

MON28335 - the monitor program of **MSK28335 DSC board** - uses the SCI interface of 'F28335 for serial communication with the PC.

The program has a 2-level structure:

1. **Low-level communication functions**, containing functions as: SCI initialization, SCI reception and transmission interrupt routines, read a character (byte) from the SCI, send a character (byte) to the SCI.
2. **High-level communication functions**, containing the monitor command interpreter. This allows you to: write data to program memory (PM) / data memory (DM), read data from PM/DM, set length of data block to read/write, execute a program from a specified PM address, customize the communication protocol, etc.

MONPC28x Monitor - PC program command interpreter, which allows you to: display PM/DM contents, fill PM/DM contents, modify PM/DM contents, evaluate DM contents, adjust DM contents, customize the communication protocol, load programs to RAM or flash memory, execute a program, etc.

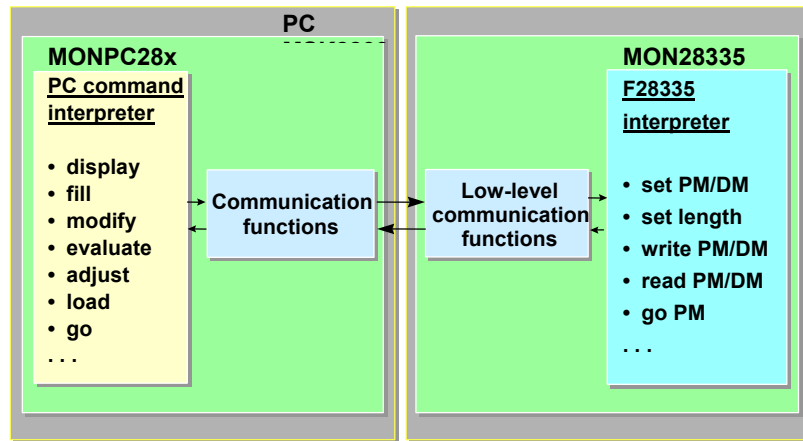


Figure 1.4. MSK28335 communication software structure

MONWIN32.DLL. All communication functions of **MONPC28x Monitor** are included in a .DLL file called **monwin32.dll**. This enables you to define your own Windows applications that would call any of the available communication functions. Thus, the integration of all communication functions with the **MSK28335 DSC board** in custom applications becomes straightforward.

1.3.2 Chip Evaluation Applications

PROCEV2833x 'F28335 chip evaluation module is a complete, fully documented collection of basic experimental programs which cover all the features of the 'F28335 DSC controller, to be used in a Digital Motion Control (DMC) structure.

Based on a user-friendly Windows environment, the application offers a quick, straightforward evaluation of the I/O features of the 'F28335 DSC controller. It contains applications for: general purpose I/O ports; A/D converters; serial interfaces (SCI and SPI); eCAN interface; watchdog and RTI; event manager components (timers, PWM, encoder interface, captures, compare units). Programs may be downloaded to the RAM memory of the **MSK28335 DSC board** and executed. All these examples are accompanied by the source code files and by ready-to-run examples which can be included into / adapted to your applications.

1.3.3 Application Development Tools

The **MSK28335** and **MCK28335 kits** include as development platform **DMC28x Developer Lite** a Windows IDE environment which offers you the possibility to develop your DSC program, through integrated project management, file editor, assembler, linker and debugger functions. The **MSK28335 kit Pro** and **MCK28335 kit Pro** include as development platform **DMC28x Developer Pro** a professional Windows IDE environment which offers assembly and C code debugging, trace functions based on real-time data acquisition, digital motor control specific plug-ins like data logger and reference generator, watch windows, control panel, etc.

For details, please refer to the user manuals of the DMC28x Developer Lite and DMC28x Developer Pro.

1.3.4 Motion Control Applications (MCK28335 kits)

DSPMOT is an integrated, graphical-environment analysis tool for DMC applications. It offers you the possibility of analyzing your DSC program variables by using on-line watches, or off-line tracking of real-time stored data. Furthermore, a point-to-point linear interpolation reference generator block may easily be included into your DSC application and its parameters may be set in the Windows environment of the **DSPMOT** program. Similar facilities may be used for standard speed/current PI controllers.

BLAC, **BLDC** and **IMVC** motion demos. Three ready-to-run digital motor control examples accompany the **MCK28335 kits**. Executed in the **DSPMOT** environment, **BLAC** and **BLDC** drive the brushless motor included in **MCK28335 kits** in AC, respectively DC modes, while **IMVC** drive the induction motor in vector control mode. Supplied also as object files, these examples enable you to replace the reference generator and/or the speed/current controllers by your own custom-defined ones. Thus, based on pre-defined motion kernel modules (real-time interrupt structure, PWM operation, current measurement, etc.), you will be able to test alternative control schemes and / or reference generators for your application.

2 Installing the MSK28335 / MCK28335 kits

This chapter describes how to install the **MSK28335 kits** or the **MCK28335 kits** on an IBM PC™ system running under Windows™ 95 / 98 / Me / NT / XP.

Both hardware as well as software procedure installation are described in successive steps.

Contents

2.1. What you'll need

2.2. Step 1: Connecting the MSK28335 DSC board to your PC

2.3. Step 2: Installing the software for MSK28335 / MCK28335 kits

2.4. Step 3: Verifying the installation

2.5. Installation errors

2.1 What you'll need

- host computer: An IBM PC or compatible, with hard disk, CD-ROM drive, running under Windows '95, 98, Me, NT 4.0 or XP, with a RS232 serial port available (or USB port + USB / RS232 converter)
- power requirements: **MSK28335 kits:** a 5 V_{DC} (±10%) power supply at minimum 0.5 A is needed to properly drive the MSK28335 DSC board alone
MCK28335 kit equipped with PM50 v3.1:
- a 9-36 V_{DC} power supply (only for the PM50 module) at minimum 1.2 A for the motor + 0.5A for the **MSK28335 DSC board**. The PM50 module will produce the necessary supply for the **MSK28335 DSC board**.
- boards: **MSK28335 kits:** MSK28335 DSC board
MCK28335 kits: MSK28335 DSC board, PM50 board
- motor: **MCK28335 kits:** 22BC-8B **escap**[®] or 3441 **Pittman** brushless motor with encoder and Hall sensors
- serial cable: DB9 plug to DB9 or DB25 socket (depending on the PC connector)
- setup CD: Containing the setup files for the kit to install: **MSK28335 kit or MSK28335 kit Pro or MCK28335 kit or MCK28335 kit Pro**
- optional hardware: An oscilloscope and/or multimeter are useful for some of the 'F28335 DSC controller evaluation applications

2.1.1 Safety information



CAUTION! THE DRIVE IS ELECTROSTATIC SENSITIVE. AVOID TOUCHING OF ELECTRONIC COMPONENTS AND METALLIC PARTS OF PCB (PRINTED CIRCUIT BOARD) !



CAUTION! BEFORE THE CONNECTING / DISCONNECTING ANY OF THE SIGNALS PLEASE TURN OFF ALL POWER SUPPLIES. ELSE SEVER DAMAGE MAY OCCUR !



CAUTION!

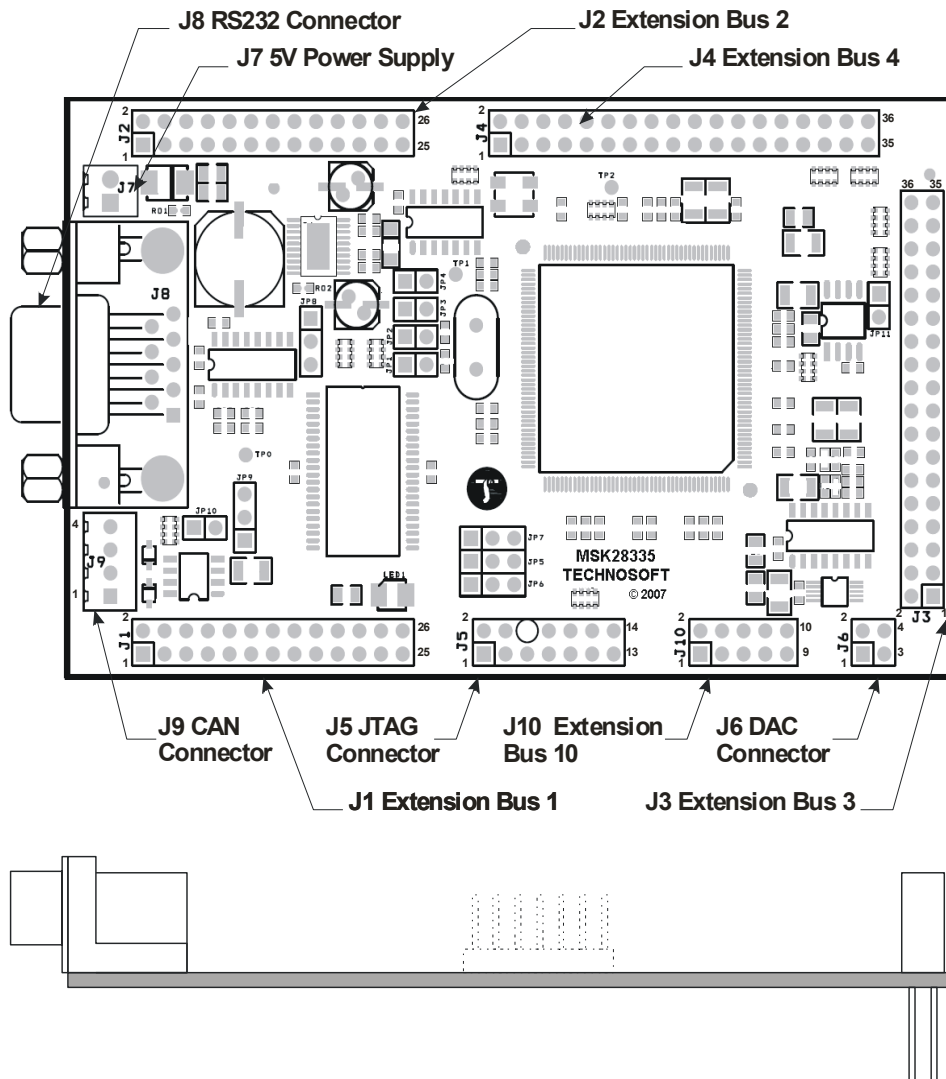
Never keep the power transistors permanently on, without commutation. In this situation the DC-bus voltage is applied directly to the motor and the higher current obtained can damage the power module and the motor. To avoid this situation please observe the following rules:

1. Do not stop the timers or disable the interrupts used for motor control, when the PWM is active. To do safely one of these operations please disable the PWM outputs before.
2. If you are working under Code Composer:
 - a. First check the code, in order to see if the used interrupts are enabled in debugging mode. See DBGIER register description in the DSC's manual.
 - b. Check the code to see if the timer's initialization allows counting during "halt" state of DSC. See TxCON register description in the DSC's manual.
 - c. Do not halt the DSC if the "Real-Time Mode" is disabled. Use "Debug | Real-Time Mode" menu command to enable this mode.
 - d. Do not place a breakpoint or probe point in interrupt code.
 - e. Do not define graphics.
 - f. Enable the "Real-Time Mode" every time you open the Code Composer.
3. If you forgot to enable the "Real-Time Mode" and the motor control program is running, do not give the "Halt" command! For stopping, shut down the motor power supply.

2.1.2 MSK28335 DSC board connections

Figure 2.1 presents the layout of the **MSK28335 DSC board**. The main components, connectors, headers and jumpers of the board are outlined.

A short description of each connector and header of the MSK28335 DSC board follows. Appendix A contains the complete functional description of the connectors and jumpers.



This drawing is not to scale 1:1

Figure 2.1. MSK28335 DSC board layout

- **Expansion bus headers (J1, J2)**

These headers provide the TMS320F28335 address and data buses and the control signals needed to add new interfaces. J1 and J2 are standard 2 x 13 pin (26-pin) 0.1" pitch headers.

Note:	1) The MSK28335 DSC board is supplied with no connector mounted on the J1 or J2 headers. In this way, depending on your own hardware structure, you can mount the appropriate connector type on the board (male/female, on the upper/lower side of the board).
-------	---

- **Expansion bus headers (J3, J4)**

These headers provide the I/O signals of the TMS320F28335 controller. J3 and J4 are standard 2 x 18 pin (36-pin) 0.1" pitch headers.

Each of these headers has all the signals needed to drive a 3-phase AC motor. A detailed description is given in Appendix A.

Notes:	1) For the MSK28335 kits , the MSK28335 DSC board is supplied with no connector mounted on the J3 or J4 headers. In this way, depending on your own hardware structure, you can mount the appropriate connector type on the board (male/female, on the upper/lower side of the board).
	2) For the MCK28335 kits , the MSK28335 DSC board is supplied with a connector mounted on the J3 for connection with the PM50 V3.1 module.
	3) Some of the I/O pins of the TMS320F28335 connected to J3 and J4 are used on the MSK28335 DSC board for driving the local interfaces (SCI serial interface, CAN interface). See Appendix A and Appendix C for detailed information.

- **Expansion bus header (J10)**

This header provides the TMS320F28335 signals, Multichannel Buffered Serial Port signals and Compare Outputs Trip signals. J10 is standard 2 x 9 pin (18-pin) 0.1" pitch headers.

- **JTAG interface header (J5)**

A 2 x 7 pin (14-pin) 0.1" pitch header, provides a standard JTAG interface making possible the connection with a JTAG emulator pad.

Note:	1) None of the Technosoft development tools for TMS320F28335 uses the JTAG interface. You don't need a JTAG pad or JTAG based software.
	2) The MSK28335 DSC board is supplied with no connector mounted on J5.

- **DAC header(J6)**

A 2 x 2 pin (4-pin) 0.1" pitch header, provides the reference and output signals for the D/A converter.

- **Power supply connector (J7)**

A 2-pin screw terminal connector is used to connect the power supply to the **MSK28335 DSC board**. Connect the 5 V_{DC} power supply with plus to pin 1 marked with “+” and minus to pin 2 marked with “-”.

Notes:	1) Do not use this connector to supply the MSK28335 DSC board when this is connected with the PM50 (default for MCK28335 kits). In this case supply only to the PM50 board. See the Appendix D
	2) Be sure to connect the supply with the right polarity.

- **Serial connector (J8)**

You need an RS-232 cable to connect your PC to your MSK28335 DSC board. The board is equipped with a standard female DB9 connector. If your PC serial connector has 9 pins, use a DB9 plug to DB9 socket straight through cable. If your PC serial connector has 25 pins, use a DB9 plug to DB25 socket straight through cable or a DB9 plug to DB9 socket straight through cable, combined with a 9-to 25-pin adapter. Appendix A shows the serial cable connections and pin assignments for both the DB9 and DB25 connectors.

Note:	Use shielded cables to ground the MSK28335 DSC board through your PC.
-------	--

- **CAN connector (J9)**

A 4-pin screw terminal connector provides the connections needed for a CAN-bus network.

2.1.3 PM50 board connections

Figure 2.2 presents the layout of the **PM50 V3.1** board. The main components, connectors, headers and jumpers of the board are outlined.

A short description of each connector and header of the **PM50** board follows. Appendix D contains the complete functional description of the pins.

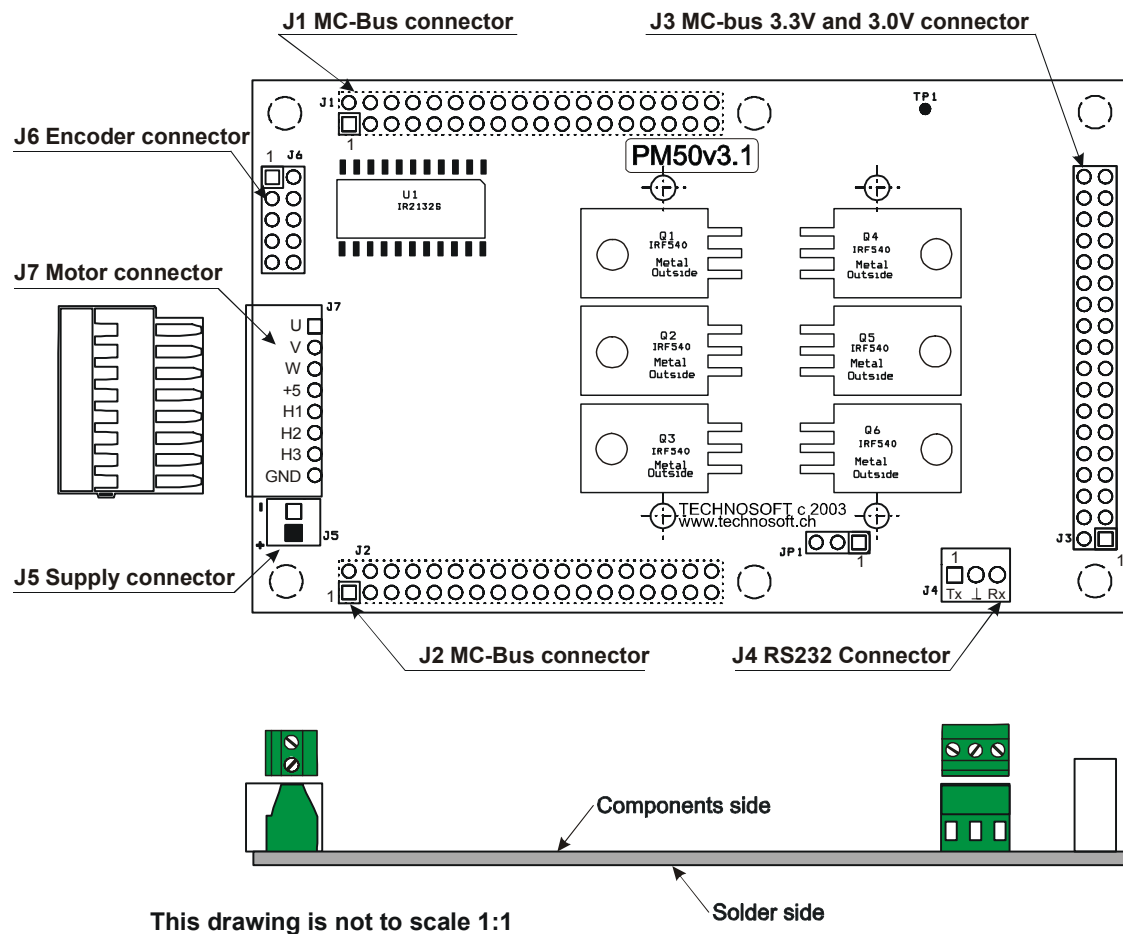


Figure 2.2. PM50 v3.1 board layout

- **MC-BUS Connectors (J3)**

Through the Motion Control Bus (**MC-BUS**) connectors the PM50 power module receives commands from the **MSK28335 DSC Board** (through J3 connector) and sends feedback signals. See Appendix D for more details

- **Motor and Hall sensors connector (J7)**

A 8-pin WAGO connector is used for motor phases and a Hall sensor interface.

- **RS-232 connector (J4)**

A 3-pin screw terminal connector for RS-232 communication. **Do not use this connector on your MCK28335 kits. Use only the 9-pin DB9 connector from the MSK28335 DSC board.**

- **Power supply connector (J5)**

A 2-pin screw terminal connector is used on the **PM50 V3.1** board, for the power supply. Connect the 9-36 V_{DC} power supply with plus to pin 1 and ground to pin 2 of the J5 connector.

Notes:	<p>1) Be sure to connect the supplies with the correct polarity (ground and positive voltages). The board is NOT protected against accidental reverse polarity connection on the power supply.</p> <p>2) The MSK28335 DSC board will receive supply through connector J3 from the PM50 board.</p>
--------	---

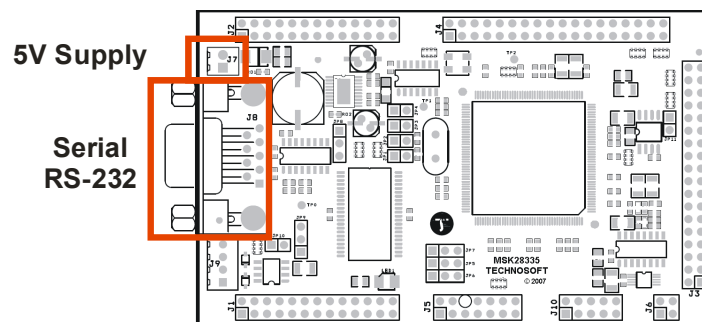
- **Encoder connector (J6)**

A 2 x 5 pin (10-pin) 0.1" pitch standard header male connector is used for an incremental encoder interface. The brushless motor supplied with the kit comes with a 10-wire flat cable with female connectors for the 500-line encoder signals.

Notes:	<p>1). Make sure to insert the flat cable correctly, i.e. with Pin 1 of the flat cable (red wire) on Pin 1 of the J6 connector!</p> <p>2). An alternate 5-pin 0.1" pitch header (J66) may be mounted on the PM50 instead of the J6 header, for other encoder connector types. J66 is pin-to-pin compatible with the 5-pin Hewlett-Packard HEDS encoders, families 550x, 554x (see Appendix D for details).</p>
--------	--

2.2 Step 1: Connecting the MSK28335 DSC board to Your PC

MSK28335 kits: Follow these steps to connect your **MSK28335 DSC board** to your PC:

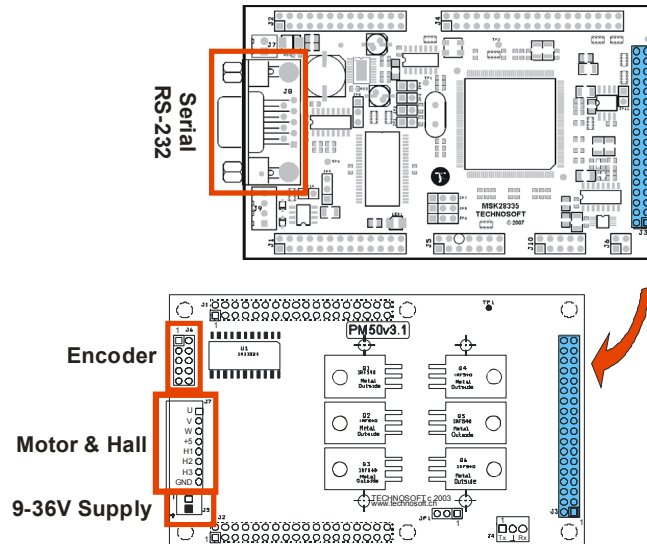


1. Switch-off the 5V power supply.
2. Connect your RS-232 cable to either of communication ports 1 and 2 on your PC. Use a 9-to-25 pin adapter if necessary.

If the PC has just USB ports and any RS-232 ports, use an USB / RS-232 converter, install the appropriate driver. Check what is the COM port allocated by Windows and set this in **MxK28335 - Control Panel -> Settings**.

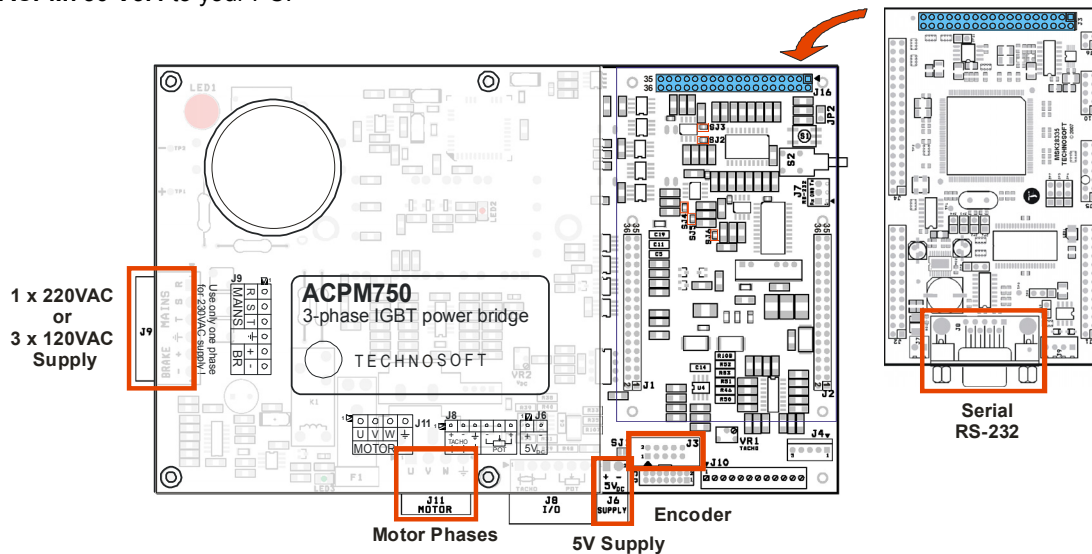
3. Plug the **RS-232** cable to the J8 connector of the MSK28335 DSC board. For details refer to paragraph A.1. MSK28335 DSC board - Connectors.
4. Connect the **5 V_{DC} power supply** to the J7 connector of the MSK28335 DSC board. For details refer to paragraph A.1. MSK28335 DSC board - Connectors.
5. Plug and switch-on the power supply. The green LED of the MSK28335 DSC board should light. If the LED remains off, please refer to par. 2.5.

MCK28335 kits with PM50 power module: Follow these steps to connect your **MSK28335 DSC board + PM50 V3.1** to your PC:



1. Switch-off the 9-36V power supply.
2. Connect your RS-232 cable to either of communication ports 1 and 2 on your PC. Use a 9-to-25 pin adapter if necessary.
If the PC has just USB ports and any RS-232 ports, use an USB / RS-232 converter, install the appropriate driver. Check what is the COM port allocated by Windows and set this in **MxK28335 - Control Panel -> Settings**.
3. Plug the **RS-232** cable to the J7 connector of the MSK28335 DSC board. For details refer to paragraph A.1. MSK28335 DSC board - Connectors.
4. Connect the **9-36V_{DC} power supply** to the J5 connector of the **PM50 V3.1** board. For details refer to paragraph D.2. Power Supply.
5. Connect the **encoder** to the J6 connector of the **PM50 V3.1** board. For details refer to paragraph D.4. Motor sensors and Appendix F. Motors Datasheets.
6. Connect the **motor phases** to the J7 connector of the **PM50 V3.1** board. For details refer to paragraph D.4. Motor sensors and Appendix F. Motors Datasheets.
7. Connect the **Hall** to the J7 connector of the **PM50 V3.1** board. For details refer to paragraph D.4. Motor sensors and Appendix F. Motors Datasheets.
8. Plug the **MSK28335** DSC board into **PM50 V3.1** using the J3 connector.
9. Switch-on the power supply. The green LED of the MSK28335 DSC board should light. If the LED remains off, please refer to par. 2.5.

MCK28335 kits with ACPM750 power module: Follow these steps to connect your **MSK28335 DSC board** + **ACPM750 V3.4** to your PC:



CAUTION! READ CAREFULLY THE ACPM750 USER MANUAL BEFORE USING!



CAUTION! DO NOT TOUCH THE ACPM750 WHEN THE LARGE RED LED IS LIT. THE ACPM750 AREA BELOW THE TRANSPARENT COVER CONTAINS HIGH VOLTAGES THAT ARE DANGEROUS FOR YOUR LIFE.

1. Disconnect **220VAC (or 120VAC) supply** and switch-off the 5V power supply.
2. Connect the **encoder** to the J3 connector of the **ACPM750 V3.4** board. For details refer to **ACPM750 User Manual** (P091.076.V34.UM.pdf) and Motor Datasheet corresponding to your motor (ASPMN00011 / SYPMN00012 / SYPMN00013).
3. *Only BRUSHLESS motor: Connect the Hall to the J10 connector of the **ACPM750 V3.4** board. For details refer to **ACPM750 User Manual** (P091.076.V34.UM.pdf) and Motor Datasheet corresponding to your motor (ASPMN00011 / SYPMN00012 / SYPMN00013).*
4. Connect the **motor phases** to the J11 connector of the **ACPM750 V3.4** board. For details refer to **ACPM750 User Manual** (P091.076.V34.UM.pdf) and Motor Datasheet corresponding to your motor (ASPMN00011 / SYPMN00012 / SYPMN00013).
5. Plug the MSK28335 DSC board into **ACPM750 V3.4** using the J3 connector.
6. Plug the **RS-232** cable to the J7 connector of the MSK28335 DSC board. For details refer to paragraph A.1. MSK28335 DSC board - Connectors.
7. Connect your RS-232 cable to either of communication ports 1 and 2 on your PC. Use a 9-to-25 pin adapter if necessary.
If the PC has just USB ports and any RS-232 ports, use an USB / RS-232 converter, install the appropriate driver. Check what is the COM port allocated by Windows and set this in **MxK28335 - Control Panel -> Settings**.
8. Connect the **5V_{DC} power supply** to the J6 connector of the **ACPM750 V3.4** board. For details refer to **ACPM750 User Manual** (P091.076.V34.UM.pdf).

9. Connect the **220VAC (1-phase) or 120VAC (3-phase) power supply** or to the J9 connector of the **ACPM750 V3.4** board. For details refer to **ACPM750 User Manual** (P091.076.V34.UM.pdf).
10. Switch-on the **5V_{DC} power supply**. The green LED of the MSK28335 DSC board should light. If the LED remains off, please refer to par. 2.5.
11. Switch-on the **1 x 220VAC (or 3 x 120VAC) power supply**. The green LED of the MSK28335 DSC board should light-on. On the ACPM750, the big and red LED1 (power) should light-on, the LED2 (error) should light-on until the software reset the error latch and the LED3 is lighted-on just about 1 sec. after connection of high voltage then is lighted-off. For details refer to **ACPM750 User Manual** (P091.076.V34.UM.pdf).

2.3 Step 2: Installing the Software for the MSK28335 / MCK28335 kits

This section explains the process of installing the software included in the **MSK28335 / MCK28335 kits** on a hard disk system. The software is provided on a CD-ROM.

1. Insert the Technosoft CD-ROM disk in the CD-ROM drive of your PC
2. The setup program starts automatically. Follow the instructions during the setup procedure. For the **MCK28335 kits**, you will be asked to specify the type of brushless motor your kit has: **Maxon / Pittman** or **escap**.

After installation, a program group will be created in your Windows environment, containing the software components of the kit and a control panel from which you can call each of the program components.

For the **MSK28335 kit**, the control panel is named **MSWIN28335** and is presented in Figure 2.3. It contains the **monitor** (MONPC28X), the **processor evaluation** (PROCEV2833x), the **DMC28X Developer Lite** development platform and the **settings** utility.

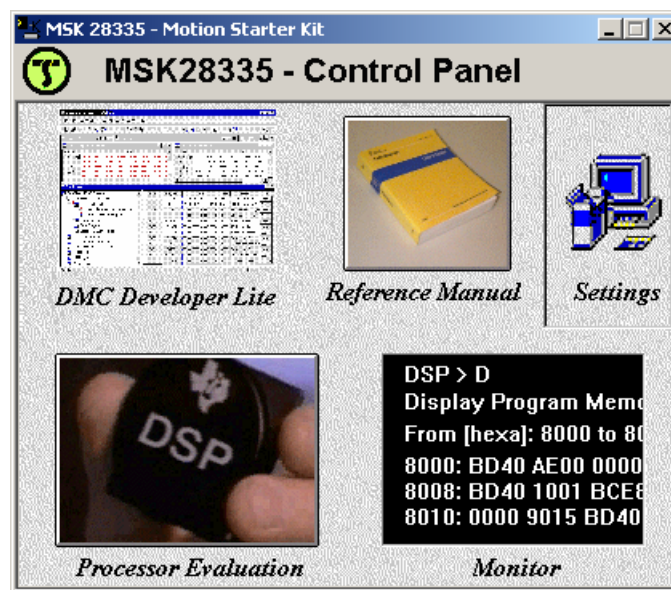


Figure 2.3. MSWIN28335 – The MSK28335 kit control panel

For the **MSK28335 kit Pro**, the general platform is **MSWIN28335 Professional** and is presented in Figure 2.4. It contains the **monitor** (MONPC28X), the **processor evaluation** (PROCEV2833x), the **DMC28X Developer Pro** development platform and the **settings** utility.

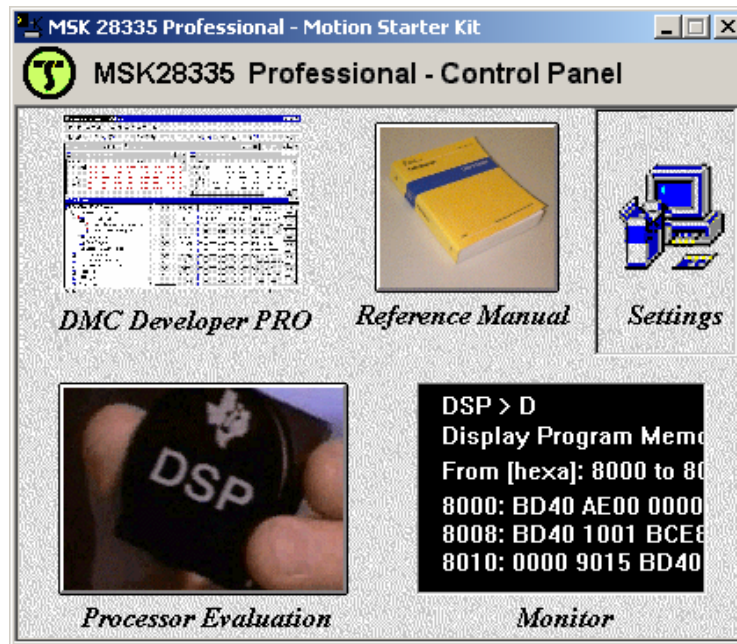


Figure 2.4. MSWIN28335 Professional – The MSK28335 kit Pro control panel

For the **MCK28335 kit**, the control panel is named **MCWIN28335** and is presented in Figure 2.5. It contains the **monitor** (MONPC28X), the **processor evaluation** (PROCEV2833x), the **DMC28X Developer Lite** development platform, the **DC brushless** and **AC brushless** motion **demos** and the **settings** utility.

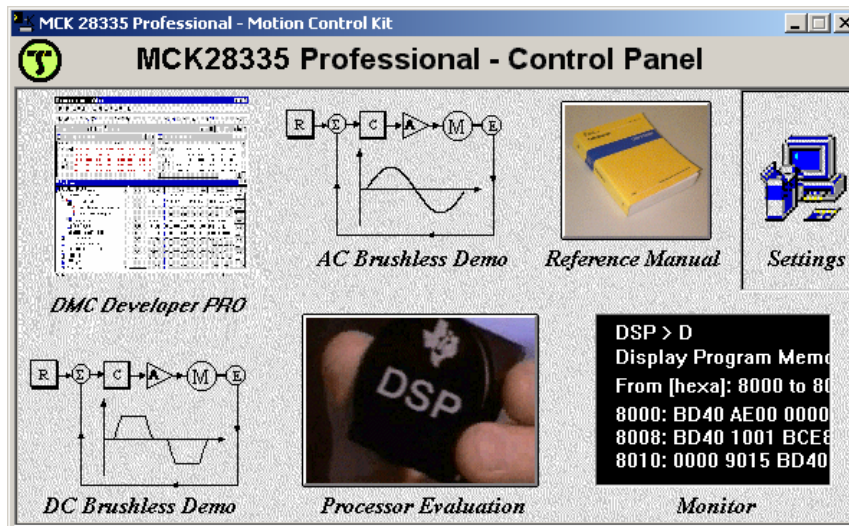


Figure 2.5. MCWIN28335 Professional – The MCK28335 kit Pro control panel

For the **MCK28335 kit Pro**, the control panel is named **MCWIN28335 Professional** and is presented in Figure 2.6. It contains the **monitor** (MONPC28X), the **processor evaluation** (PROCEV2833x), the **DMC28X Developer Pro** development platform, the **DC brushless** (BLDC) and **AC brushless** (BLAC) motion demos and the **settings** utility.

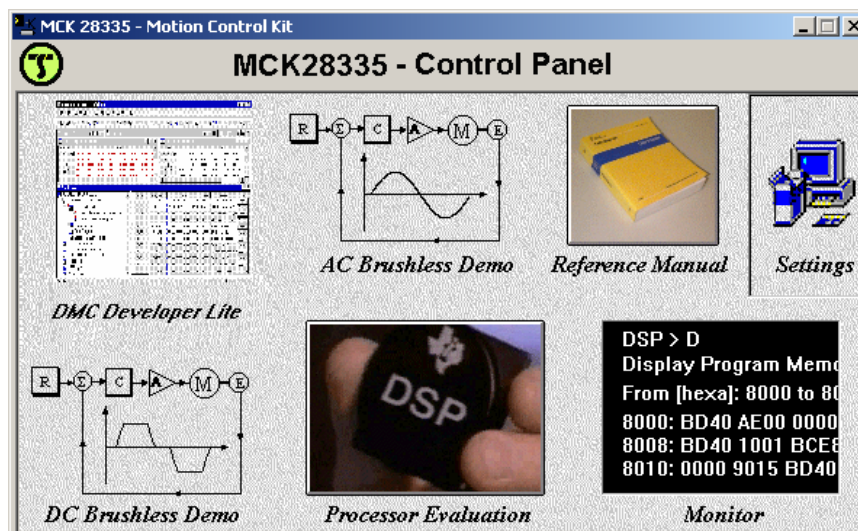


Figure 2.6. MCWIN28335– The MCK28335 kit control panel

2.4 Step 3: Verifying the Installation

The following procedure shows you how to check the serial communication between the MSK28335 DSC board and your PC. Perform the following operations:

1. In Windows 95/98 or NT select
 - For **MSK28335 kit** : "**Start | Programs | MSWIN28335 | MONPC28X**"
 - For **MSK28335 kit Pro**: "**Start | Programs | MSWIN28335 Professional| MONPC28X**".
 - For **MCK28335 kit** : "**Start | Programs | MCWIN28335 | MONPC28X**"
 - For **MCK28335 kit Pro**: "**Start | Programs | MCWIN28335 Professional| MONPC28X**".
2. The window from Figure 2.7 should be displayed on your screen. In this case, the monitor program is communicating properly with the **MSK28335 DSC board**.

You may enter a simple command as "display from data memory, from address 200h to 210h" using the following instructions:

```
DSP> d<CR>
      Display data memory
      From [hexa]: 9000<SP> to 9010<CR>
```

and get the contents of these memory locations displayed. In this case, both basic hardware and software are correctly installed, and you may proceed to the execution of all the other software components of the package. Refer to Chapter 3 for details on these programs.

Remark: Use the Quit command (**q<CR>**) to exit from the monitor program.

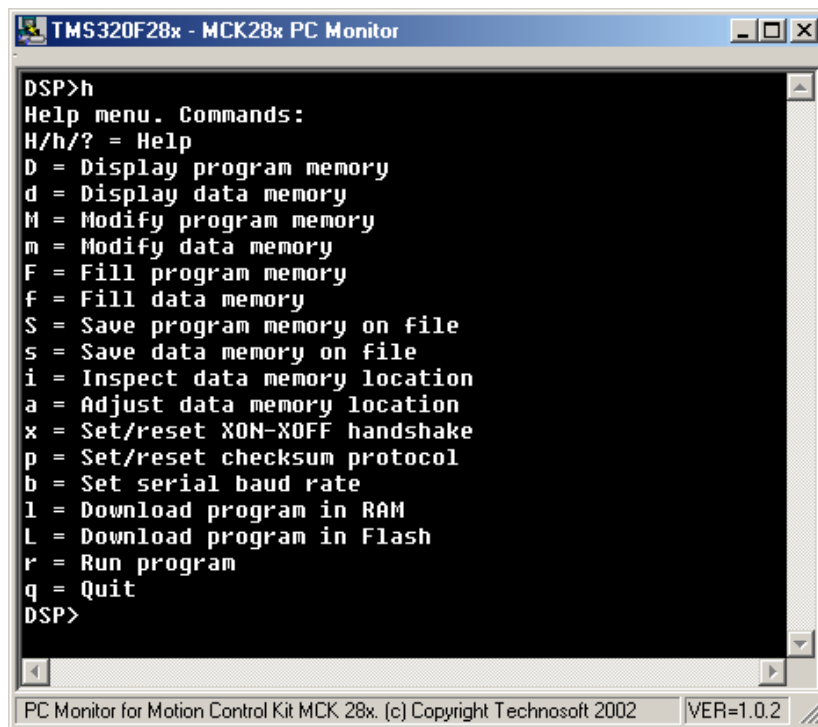


Figure 2.7. Monitor screen for proper operation of the communication between the **MSK28335 DSC board** and the PC

If on starting the monitor program you get, instead of the screen from Figure 2.7, a **"Communication Error!"** message, the most probable cause is an incorrect setting of the communication channel.

Close the monitor screen, and then start the **"Settings"** utility program. Change the serial connection port, press the **OK** button, and then try executing the **Monitor** program again. If the communication error persists, see the next section of this chapter.

2.5 Installation errors

If you get a **"Communication error"** message when trying to execute the **"Monitor"** program, or one of the other software components of the **MSK28335 kit**, one or more of the following conditions may be the cause:

- ✓ you have used an incorrect communication port. Refer to Section 3.2 for settings on the communication port.
- ✓ your communication port channel may be interrupted or noisy. If so, check the serial cable and connectors on the PC, as well as those on the **MSK28335 DSC board**.
- ✓ a mouse driver or other software may be already using the communication port you have attempted to use with the **MSK28335 DSC board**. If so, try another communication port. Refer to Section 3.2 for settings on the communication port.
- ✓ your power supply is not properly connected to the **MSK28335 DSC board (MSK28335 kits)** or to the **PM50 board (MCK28335 kits)**, or is not supplying the required voltage. The green LED of the **MSK28335**

board should become ON if the board itself is properly supplied and operating. Try to press the “**RESET**” button of the **MSK28335 DSC board** and try to restart the program.

3 Motion Control Evaluation and Development Tools

This chapter describes the basic structure of the software components included in the **MSK28335 kits** and **MCK28335 kits**. The chapter gives an overview of all the components of the package.

Contents

- 3.1. The MSWIN28335 and MCWIN28335 platforms**
- 3.2. Communication programs**
- 3.3. Processor evaluation**
- 3.4. Application development toolbox (*MSK28335 kits*)**
- 3.5. Motion analysis tool (DSPMOT) (*MCK28335 kits*)**
- 3.6. Motion demos (BLAC, BLDC) (*MCK28335 kits*)**
- 3.7. MSWIN28335 / MCWIN28335 platforms settings**

3.1 The MSWIN28335 and MCWIN28335 platforms

The software of the **MSK28335** and **MCK28335 kits** consists on a package of several programs for the **MSK28335 DSC board** and for the PC computer connected to the DSC board.

These applications are grouped, at the PC level, on a control panel Windows platform called:

- **MSWIN28335** for the **MSK28335 kit**
- **MSWIN28335 Professional** for the **MSK28335 kit Pro**
- **MCWIN28335** for the **MCK28335 kit**
- **MCWIN28335 Professional** for the **MCK28335 kit Pro**

These Windows applications communicate with specific DSC code implemented at the **MSK28335 DSC board** level.

MSWIN28335 and **MSWIN28335 Professional** control panels platforms offers you the possibility to perform basic communication with the board, to examine all the basic I/O features of the DSC controller, or to develop and test your own applications.

MCWIN28335 and **MCWIN28335 Professional** control panels platforms offers you the possibility to perform basic communication with the board, to examine all the basic I/O features of the DSC controller, to develop and test your own applications or to execute and graphically evaluate the included motion demos applications

After the setup, you can open the control panel platform (see Figures 2.3 to 2.6) and from here you can run any component with a double-click on the component icon. Basically, these platforms contain the following modules:

Basic Communication

Communication module including:

- **MON28335** - TMS320F28335 monitor program for the **MSK28335 DSC board**
- **MONPC28X** - PC monitor program for communication between **MSK28335 DSC board** and PC
- **MONWIN32.DLL** - communication module with the **MSK28335 DSC board**, ready to be integrated in user developed Windows applications.

3.1.1 Processor Evaluation

Processor evaluation based on ready-to-run applications – **PROCEV2833x**, containing ready-to-run examples for the evaluation of all the basic I/O peripherals of the TMS320F28335 DSC.

Application Development Basic Platform (*MSK28335 kit and MCK28335 kit*)

- Windows IDE for motion applications development and testing – **DMC28X Developer Lite**, including :
 - Integrated assembler and linker
 - Incorporated assembler-level debugger
 - Memory and I/O registers view / modify
 - Integrated assembler source code editor
 - Watch windows
 - Advanced project management

Applications Development Professional Platform (MSK28335 kit Pro and MCK28335 kit Pro)

- Windows IDE for motion applications development and testing – **DMC28X Developer Pro**, including all the features of the DMC28X Developer Lite plus:
 - Possibility to work with Texas Instruments assembler, linker C compiler
 - C-level debugger
 - Possibility to integrate a data logger module
 - Possibility to integrate a reference generator module
 - Control panel for easy parameter settings and global application status evaluation

Motion Control Applications (MCK28335 kit)

- Advanced graphical tool for motion applications development and evaluation (**DSPMOT**) including :
 - Graphical reference generator
 - Digital setting of controller parameters
 - Automatic download of motion parameters to **MCK28335**
 - Motion data logger and loader
 - User definable reference and controller structure through C and / or assembly language programming
 - Possibility to easily compare the performances of different structures
- **BLAC/BLDC** - TMS320F28335 programs for AC and DC brushless motor speed control. Features :
 - Included in and taking advantage of all the **DSPMOT** program environment features
 - Digital implementation of PI current and speed controllers
 - Ideal examples for PM brushless user-developed motion applications

Due to their structure, these control panels represent basic evaluation and development Windows platforms for the **MSK28335 / MCK28335 kits** configurations

At the basic level, they offer the **PROCEV2833x** processor evaluation applications. These programming examples are of great value for users, providing them with a starting point for their applications.

With **MONPC28X Monitor** facilities, you can easily download and execute a DSC program.

At the advanced level, you can create, execute and test your own application. The development of user specific DSC software is facilitated by the easy integration in the **DMC28X Developer Lite** or **Pro** frameworks. This allows you to use the advanced debugging features of these development platforms, which provide rapid debugging and implementation of real-time control applications.

In the **MCK28335 kits**, the **BLAC/BLDC** motion control demo programs are ready-to-run solutions for AC and DC brushless speed control. These examples offer the possibility of analyzing a basic motion control structure. Different shapes of motion reference and controller parameters may be defined. During run-time, the main drive system parameters like speed reference, motor speed/position, speed error, current command(s) and motor current(s) are saved in external data memory. The system behavior can be evaluated by using the advanced graphical module included in the **DSPMOT** program.

At the advanced level, **MCWIN28335 kits** allows you to define and integrate your own reference and/or controllers written in C or assembly languages, within the two motion applications.

The development of user specific software for motion control is facilitated by the easy integration in the **DSPMOT** framework. This allows you to use the advanced analysis features of **DSPMOT**, which based on real-time communication, offers you facilities for quick debugging and evaluation of the control applications.

All the control panel platforms are fully compatible with Texas Instruments development tools (compiler, assembler, linker).

3.2 Communication Programs

A click on the **Monitor** symbol present in all the control panel platforms will activate the **MONPC28X** program. This program allows the communication with the corresponding **MON28335** monitor program from the **MSK28335 DSC board**. You may perform basic operations as display, modification, fill of data/program memory, set communication parameters, download programs in the RAM memory, execute a program, etc. Low-level as well as high-level monitor functions may be called from your applications. The corresponding PC monitor functions, grouped in the **MONWIN32.DLL** file, may also be included in your Windows-based applications using the **MSK28335 DSC board**. See chapter 4 for details about these programs and their components.

3.3 Processor Evaluation

A click on the **Processor Evaluation** symbol present in all the control panel platforms will activate the **PROCEV2833x** program. This program offers the possibility to execute simple applications on the **MSK28335 DSC board**, allowing you to program and test the basic I/O functions of the DSC chip (as timers, ePWM, eCAP, QEP, GPIO, SCI, SPI, I2C, McBSP, CAN, watchdog, A/D converters). Executed from a Windows environment, these basic applications are fully documented. The source files of each application project on the DSC may be examined by you, and integrated in your own specific applications. Moreover, for each function analyzed, a corresponding ready-to-run DMC28X Developer Lite/Pro project example is available. See chapter 5 for details about this program and its components.

3.4 Application Development Platforms

A click on the **DMC28X Developer Lite** symbol present in the **MSWIN28335** and **MCWIN28335** control panel platforms will activate the **DMC28X Developer Lite** program. A click on the **DMC28X Developer Pro** symbol present in the **MSWIN28335 Professional** and **MCWIN28335 Professional** control panel platforms will activate the **DMC28X Developer Pro** program. These development platforms represent an advanced tool for DSC programs development and analysis, allowing you to handle, in an advanced Windows IDE, the DSC applications. The platforms allow you to operate in a very efficient DSC project management environment. The projects may contain several assembler files, which may be edited using the incorporated advanced source editor. Then, the assembler (C compiler for DMC28X Developer Pro) and linker will generate an executable file, in the COFF standard format. The executable file may be downloaded on the DSC board. Using the debugging functions, you can then execute the code using step-by-step, or breakpoint operations. Memory, I/O registers and internal registers can be examined and/or modified. Advanced watch functions are also included. The DMC28X Developer Pro also includes data logger and reference generator modules that can be easily integrated into your code, providing an efficient solution during application development and testing. See the **User Guides** (included in the kits) for the DMC28X Developer Lite and DMC28X developer Pro for details regarding these products.

3.5 Motion Analysis Tool (DSPMOT) (*MCK28335 kit*)

The **DSPMOT** program is installed for the **MCK28335 kit** and **MCK28335 kit Pro**. It is automatically called when the BLAC and BLDC demos are open. It can also be open with "**Start | Programs | MCWIN28335 | DSPMOT**" Windows command for **MCK28335 kit** or "**Start | Programs | MCWIN28335_Professional | DSPMOT**" Windows command for **MCK28335 kit Pro**. **DSPMOT** program represents an advanced tool for digital motion control programs analysis and evaluation. It allows you to download applications to the **MSK28335 DSC board**, examine/set the value of program variables, execute the DSC program. By calling from your program appropriate functions (supplied in a DSC library), you may activate communication, data logging and linear reference functions in your application. Thus, all the powerful Windows-based graphical

features of **DSPMOT** program may be used, and advanced debugging of the program may be performed, significantly reducing the development time of the motion control applications. See chapter 6 for details about this program.

3.6 Motion Demos (BLAC / BLDC) (*MCK28335 kit*)

A click on the **Brushless AC demo** button in the **MCWIN28335** or **MCWIM28335 Professional** control panel platforms will activate the **BLAC** motion demo. Executed in the **DSPMOT** program environment, this example implements a speed control loop for a three phase brushless motor, driven in AC mode. You may define the PI discrete current (i_d and i_q) and speed controllers parameters, as well as the speed reference shape. See chapter 7 for details about this program.

A click on the **Brushless DC demo** button in the **MCWIN28335** or **MCWIM28335 Professional** control panel platforms will activate the **BLDC** motion demo. Executed in the **DSPMOT** program environment, this example implements a speed control loop for a three phase brushless motor, driven in DC mode. You may define the PI discrete current and speed controller's parameters, as well as the speed reference shape. See chapter 7 for details about this program.

3.7 Control Panel Platform Settings

A click on the **Settings** symbol in any of the control panel platforms will activate the settings dialog, which allows you to define some general parameters. In this dialog (see Figure 3.1), you can:

- select the communication channel (**COM1** to **COM4**) of the PC, connected to the **MSK28335 DSC board**
- select the serial communication speed (**9600bps** | **19200bps** | **38400bps** | **56000bps** | **115200bps**)
- set/reset **Xon-Xoff** protocol
- set/reset **checksum** protocol
- set the communication timeouts parameters
- set the value of the voltage used to supply the motor (needed to compute some display parameters used in the **PROCEV2833x** program) (setting valid only for the **MCK28335 kits**)
- set the number of lines of the incremental encoder (needed for the correct execution of the motion demos **BLAC** and **BLDC**, as well as for the **PROCEV2833x** program) (setting valid only for the **MCK28335 kits**)

Note: after changing the settings, you must close and restart the component applications from the control panel platform, so that the new settings have effect.

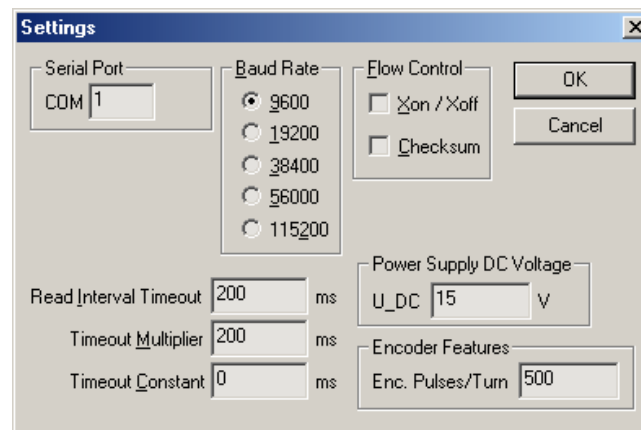


Figure 3.1 Settings dialog from control panel platforms

4 Communication Module

This chapter describes the structure and functions of the different program modules used for communication between the **MSK28335 DSC board** and the PC computer.

Contents

- 4.1. Monitor concept**
- 4.2. Communication module description**
- 4.3. Using the PC monitor – MONPC28X**
- 4.4. Using the MSK28335 DSC board monitor - MON28335**
- 4.5. Flash Programming Tool**
- 4.6. Creating a customized PC monitor using MONWIN32.DLL**

4.1 Monitor Concept

Users eager to run their applications on a new DSC system are often disappointed by the lack of an immediate solution to classical problems like:

- How to download a program
- How to check if a program works OK
- How to debug a program if it doesn't work as expected

What users need is a tool helping them to:

- download their program from a host computer
- upload for visualization the results
- inspect or modify DSC program and data memory locations

Put in other words, what users need is a communication program through which to exchange data between the DSC system and the host computer in order to perform the operations mentioned above. Such a program is often called a **monitor**.

A well-designed monitor offers not only a fixed solution but also a package of communication functions easy to integrate in users programs. This gives users the possibility to focus from the beginning on their applications. The lack of a good monitor oblige the users to spend an annoying extra time to create their own communication routines. In the particular case of motion control applications the necessity to provide a monitor is even stronger. Here data should be exchanged in parallel with real-time control operations and designing reliable DSC communication routines is far from being an easy task.

Note: The monitor can be obtained under license for use without the MSK28335 DSC board.

4.2 Communication Module Description

All the kits is delivered with a communication module that offers a robust solution for the communication between a PC and the **MSK28335 DSC board**. The communication module includes:

- **MON28335** - the 'F28335 DSC controller monitor program. Written in Flash EEPROM memory, this is the default program that executes after reset, if the boot mode is selected to flash. It includes a set of communication routines that may also be called from your application DSC programs. The most important of these routines is the command interpreter, which accepts and executes a set of elementary commands (see par. 4.4. for details).
- **MONPC28X** - a PC monitor. This program offers you a large set of monitor functions including those named in par. 4.1. **MONPC28X** commands are implemented using the basic PC communication functions defined in **MONWIN32.DLL**.
- **MONWIN32.DLL** - a library containing basic PC communication functions. These functions translate the PC monitor commands into a set of elementary commands, which are recognized and executed by **MON28335** command interpreter at DSC level.

The combination between **MON28335** at DSC level and **MONPC28X** at PC level, offers an immediate, ready to run solution for PC-**MSK28335 DSC board** communication. In the same time, by offering access to the PC communication library **MONWIN32.DLL** and to the DSC communication functions stored in the Flash EEPROM memory of the 'F28335 DSC controller, the communication module offers the possibility to create customized communication solutions.

The easiest way to work with the communication module is, at PC level, to use **MONPC28X Monitor**, and to include calls to the **MON28335** command interpreter at the DSC level (see par. 4.4. for details). Thus,

MONPC28X commands may be interpreted and executed in a transparent way for you, step by step, each time when **MON28335** command interpreter is called.

Another possibility is to design a customized PC monitor using the communication functions from **MONWIN32.DLL** (see par. 4.5. for details). As in the previous case, at DSC level the only requirement is to include calls to **MON28335** command interpreter in your DSC application program.

A third possibility is to extend the communication functions of the DSC program by adding a second command interpreter containing specific user commands. This second command interpreter can work in parallel with **MON28335** command interpreter so that all the facilities offered by **MON28335** command interpreter can be preserved (see par. 4.4. for details)

Remark: The use of **MON28335** communication functions, which are written in Flash EEPROM memory, is not mandatory. You can easily bypass these routines and define your own DSC communication functions. However in this case the compatibility with **MONPC28X** program or **MONWIN32.DLL** functions is lost. Implicitly, you will not be able to use other programs of the control panel platforms, as **DMC28X Developer Lite/Pro**, **PROCEV2833x**, **BLAC** and **BLDC** demos using **DSPMOT**.

4.3 Using the PC Monitor - MONPC28X

MONPC28X is a PC program that communicates with the **MSK28335 DSC board** through a serial asynchronous interface (COM1 to COM4). The serial communication uses 8 data bits, no parity and 1 stop bit for baud rates below or equal with 38400 or 2 stop bits for baud rates over 38400. The baud rate can be set from 9600 up to 115200 baud with or without XON-XOFF handshake, with or without checksum & acknowledge protocol. For DB9 PC connectors the serial link requires a standard straight through (one to one) male to female cable. Shielded cables are recommended.

MONPC28X offers a set of monitor commands that allows you to:

- download a program in the **MSK28335 DSC board** RAM memory
- upload and save in a file the contents of program or data memory
- start execution of a program
- display the contents of program or data memory locations
- modify the contents of program or data memory locations
- inspect and adjust data memory contents while running a program
- customize serial communication between the PC and the **MSK28335 DSC board**

4.3.1 Invoking MONPC28X

MONPC28X may be invoked from the control panel platforms by clicking on the **Monitor** icon. When **MONPC28X** is executed as a standalone program it can be invoked with 4 optional parameters in the command line:

```
MONPC28X [COM1|COM2|COM3|COM4] [9600|19200|38400|56000|115000] [X] [P]
```

MONPC28X	is the command invoking the PC monitor
COM1 COM2 COM3 COM4	selects which PC serial port to be used (default: COM2)
9600 19200 38400 56000 115000	selects the baud rate (default: 9600)
X	uses XON-XOFF handshake (default: no handshake)
P	uses checksum protocol (default: no checksum)

The command line parameters are not case sensitive.

When **MONPC28X** is invoked from control panel platforms, the **communication channel** is automatically set according with the platform settings. The other parameters take the default values.

4.3.2 Operating Modes

When **MONPC28X** is started, it first checks the communication with the **MSK28335 DSC board**. If the check passes, the communication settings are settled according with the command line parameters. **MONPC28X** enters in the **normal operating mode** and displays the following menu:

```
Help menu. Commands:
H/h/? = Help
D = Display program memory
d = Display data memory
M = Modify program memory
m = Modify data memory
F = Fill program memory
f = Fill data memory
S = Save program memory on file
s = Save data memory on file
i = Inspect data memory location
a = Adjust data memory location
x = Set/reset XON-XOFF handshake
p = Set/reset checksum protocol
b = Set serial baud rate
l = Download program in RAM
L = Download program in Flash
r = Run program
q = Quit
DSP>
```

If the communication check fails, an error message is displayed and **MONPC28X** execution is aborted.

When one of the letters from the menu is pressed, the corresponding command is selected. At this point, the user has 2 options:

- press **ENTER** to validate the command. In this **MONPC28X** displays the command name then asks the command parameters. This procedure is recommended at the beginning till you are getting used to the command syntax and is presented in detail in par. 4.3.4.
- continue with the command parameters. This procedure is recommended when you are already familiar with the command syntax.

If the “u” alphanumeric key is selected, followed by **ENTER**, the monitor enters in the **user terminal operating mode**. In this mode everything typed is transmitted to **MSK28335** board (the ASCII code) and anything received from the DSC board is displayed on the PC screen. **MONPC28X** remains in the terminal mode until **ENTER** is pressed.

You can easily differentiate the two operating modes of **MONPC28X** from the message displayed on the last line - the information line of the window.

When **MONPC28X** is in normal operating mode the displayed message is:

PC Monitor for Motion Starter Kit MSK28335. (c) Copyright Technosoft 2003

When **MONPC28X** is in user terminal operating mode the displayed message is

PC Monitor works as terminal

4.3.3 Using the Built-in Editor

Most **MONPC28X** commands ask you to introduce data parameters. For example the command **Display from data memory** asks the start address and the stop address of the data memory block to be displayed. The built-in editor offers the following facilities:

- data digits can be typed and deleted as in any text line editor. A data value is taken into consideration only after a **SPACE**. After last parameter is typed, the command is validated with **ENTER** or **SPACE**;
- if during a data editing a dot '.' or **ESC** is typed, the command is aborted;
- all commands except the **Modify program/data memory** commands are also aborted if **ENTER**, or **SPACE** is pressed when no other data digit is introduced. In the modify commands case, typing **ENTER**, or **SPACE** without introducing a data lets the current memory location unmodified and jumps to the next one.

The data introduced by you are interpreted as **hexadecimal** numbers (except the baud rate selection). Maximum number of digits is 4. The **a-f** digits can be both lowercase or uppercase. All data displayed by **MONPC28X** (except the current baud rate) are also in hexadecimal format.

4.3.4 Monitor commands

H/h/? - displays the help menu

D/d - displays program/data memory contents between a start address and a stop address

Example 4.1: display the data memory contents, between addresses 9000h and 9010h

```
DSP>d<CR>
Display data memory
From [hexa]: 9000<SP> to 9010<SP>
009000: 0000 0000 0000 0000 0000 0000 0000 0000
009008: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
DSP>
```

Notes:	1) . The characters in bold are the characters introduced by you
	2) . <CR> = ENTER and <SP> = SPACE keyboard keys

M/m - modifies program/data memory contents at successive addresses

Beginning with a start address, the command displays the memory location address and its contents. You are asked to type a new value, which will replace the location contents. If you type just **ENTER** or **SPACE** the memory location remains unmodified. Then the next address and its contents are displayed and the process is repeated until you end the command with a dot '.' or **ESC**.

Example 4.2: Modify the data memory starting from the address 9009h

```
DSP>m<CR>
  Modify data memory
  From [hexa]: 9009<CR>
009009: 005F New value: 25A7<CR> // (9009) = 25A7h
00900A: 005D New value: <CR>    // (900A) = 5Dh
00900B: 035C New value: .        // abort the "modify" command
DSP>
```

F/f - fills program/data memory between a start address and a stop address with a value

Example 4.3: Fill the program memory contents from 9000h to 9010h with 5555h

```
DSP>F<CR>
  Fill program memory
  From [hexa]: 9000<SP> to 9010<SP> with value [hexa]: 5555<SP>
DSP>
```

S/s - saves program/data memory contents between a start address and a stop address on a file

Example 4.4: Save the data memory contents from 9000h to 9010h, on the file "savefile.dat"

```
DSP>s<CR>
  Save data memory
  From [hexa]: 9000<SP> to 9010<SP>
  to file: savefile.dat<CR>
DSP>
```

i - continuously reads a data memory location from a given address and display its contents

This command may be used during the execution of your application, in order to examine the modification in time of a specific variable of the program.

Example 4.5: Inspect the contents of the data memory from the address 9000h

```
DSP>i<CR>
  Inspect data memory
  From [hexa]: 9000<SP>
009000: 01A4 // end the command by pressing any key of the PC keyboard
DSP>
```

a - continuously adjust the contents of a data memory location from a given address

This command may be used during the execution of your application, in order to modify "on the fly" the value of some parameters (controllers' coefficients, for example). Use the "LEFT" and "RIGHT" keys in order to position the cursor on one of the four hexadecimal digits, and "UP" or "DOWN" keys in order to increase, respectively decrease by a unit the selected digit.

Example 4.6:

```
DSP>i<CR>
  Adjust data memory
  From [hexa]: 9000<SP>
009000: 01A4 // with left / right arrows change the underscored editable digit
```

```
DSP>                                     // with up / down arrows add / subtract 1 from underscored digit
                                     // press <CR> to exit
```

x - enables or disables the XON-XOFF handshake during serial data exchange

Example 4.7: enable the XON-XOFF handshake protocol

```
DSP>x<CR>
XON-XOFF handshake [1:enabled/0:disabled]
Actual = 0 New value: 1<CR>
XON-XOFF handshake is enabled
DSP>
```

p - enables/disables MSK28335 checksum & acknowledge protocol

When this protocol is enabled, each group of data sent to **MSK28335 DSC board** is followed by a checksum. After reception of the entire group, the DSC board sends an acknowledge character to confirm the correctness of the data. The same procedure is used when groups of data are sent from the **MSK28335 DSC board** to the PC. This protocol detects communication errors and tries to solve them. If it can't, it issues a communication error message. This protocol guarantees the integrity of the exchanged data if no error message is displayed. However it reduces the transfer speed with about 30%.

Example 4.8: Enable the checksum protocol

```
DSP>p<CR>
Checksum protocol [1:enabled/0:disabled]
Actual = 0 New value: 1<CR>
Checksum protocol is enabled
DSP>
```

b - sets/changes serial communication baud rate

The baud rate change is done simultaneously on the PC and on the **MSK28335 DSC board**.

Example 4.9: Set the baud rate at 19200 baud

```
DSP>b<CR>
Set baud rate: 9600 | 19200 | 38400 | 56000 | 115200
Actual = 9600 New value: 19200<CR>
DSP>
```

l (lower case L) - downloads a DSC program in the internal or external RAM memory

A file in standard COFF format (extension .out) may be downloaded into the RAM memory of the DSC system.

Example 4.10: Download the "myfile1.out" file in the RAM memory

```
DSP>l<CR>
Download COFF object file in RAM memory
File name (and extension): myfile1.out<CR>
OABC                                     // shows progress of program download
Program downloaded OK in RAM memory
Entry point = 3F8080
DSP>
```

Notes:	1). The downloaded file must be located (at the link stage) into a valid address range of the RAM memory (between 200000h to 23FFFFh for the external RAM memory, or between
--------	--

	000400h to 00007FFh, or 008300h to 009FFFh, Or 3F8080h to 3F9FFFh in the internal memory - see also par. 4.4.8).
	2). The '1' monitor command may be used to write a program in the RAM memory. The command doesn't start the program execution. One has to execute the run command (see below) in order to execute the downloaded program. As the start address one must use the entry point returned by the '1' command.
	3). During download in RAM, XON-XOFF handshake and checksum & acknowledge protocol are enabled to guarantee the data integrity. After download, the initial settings are restored. These modifications are done automatically and are transparent for you.

r - runs a program (previously downloaded into the RAM program memory)

Example 4.11: Execute a program that starts at address 9050h

DSP>**r**<CR>

Start address [hexa]: **3f8500**<CR>

DSP> *// the monitor program executes a branch to the start address*

q - quits **MONPC28X** monitor

When this command is called, it first restores the default settings of the **MSK28335 DSC board** - PC serial communication (baud rate 9600, 8 data bits, 1stop bit, no parity, without the XON-XOFF handshake, without the checksum & acknowledge protocol) and only then closes the monitor.

The default settings are the initial values, set by **MONPC28X** when it starts and by **MON28335** on the DSC board after reset or power-up. At start-up **MONPC28X** checks the PC - **MSK28335 DSC board** communication using the default settings and only then modifies them according with the command line parameters.

By restoring the default settings before exit, **MONPC28X** guaranties that on next call the PC and **MSK28335** settings will be synchronized both when:

- the **MSK28335 DSC board** was reset/powered-up
- the **MSK28335 DSC board** was not reset and keeps the last settings set by **MONPC28X**

4.4 Using the MSK28335 monitor - MON28335

MON28335 is a DSC monitor program that occupies the sector B and first 2 words of the sector A of the 'F28335 DSC controller internal Flash EEPROM memory. After reset or power-up, **MON28335** is the first program executed if the following jumpers configuration meet:

JP1	SHORT
JP2	OPEN
JP3	OPEN
JP4	OPEN

The main purpose of **MON28335** is to provide a set of communication functions through which data can be sent to / received from the outside world. The goal is to let you to exchange data between your programs and another device by simply calling the communication functions defined in **MON28335**.

Apart from this main function, **MONPC28X** does other important job, a default system initialization.

4.4.1 System initialization

MON28335 does the following initializations:

- Watchdog counter: reset and disabled (you can enable it later in your application)
- 'F28335 DSC controller clock module: multiply 5 times external quartz frequency. The quartz used on the **MSK28335 DSC board** has 30 MHz, hence the CPUCLK is set at 150 MHz.
- Enable SCI-A, XINTF and GPIO clocks
- Disable interrupts
- Initialize PIE registers
- Initialize PIE Vector Table
- Set the RAM and flash wait-states
- Initialize SCI-A module for monitor protocol
- Light on the led

Notes:	1.) After reset, MON28335 initializes data memory locations for all interrupts. MON28335 uses the RXINT, TXINT and TRAP interrupts. The associated ISRs addresses, defined in MON28335 , are stored in the corresponding data memory locations. For other ISRs, MON28335 puts in the table the address of a dummy ISR. This ISR does nothing but the context restore.
	2.) MON28335 sets RXINT and TXINT interrupts with low priority i.e. as INT9. Except this interrupt, MON28335 masks all the other interrupts. It is yours program task to unmask the other interrupts it needs.

4.4.2 Hierarchical Organization

MON28335 uses communication functions organized in 3 hierarchical levels.

At the lower level, there are the basic functions used to perform PC - **MSK28335** communication. These functions initialize the 'F28335 DSC controller serial communication interface (SCI), set interrupt vectors for serial reception and transmission and implement the ISRs for SCI reception and SCI transmission interrupts.

At the intermediate level there are 2 functions: one reads data from a reception buffer and the other sends data to SCI.

At the higher level there is a command interpreter which identifies and performs monitor commands like those presented in par. 4.3.

The function "**main**" of the **MON28335** monitor is just an infinite loop that calls the command interpreter.

4.4.3 Using the MON28335 Functions

You have access to three of the **MON28335** functions. These are:

int read_char (void) - reads the oldest character/byte received from the reception buffer

Call address: **0x00330002**

Returns: the data received in the 8LSB and 00h in the 8MSB, or
100h if the data was received with errors, or
200h if the reception buffer is empty.

int send_char (int) - sends a character/byte to the SCI

Call address: **0x00330004**

Returns: 1 if the transmission was done
0 if the transmission was blocked after receiving an XOFF character.

If the XON-XOFF handshake protocol is disabled it always returns 1.

int cmd_interpreter (void) - executes the commands received in a transparent way for the user

Call address: **0x00330000**

Returns: 0 when it detects the user data command code (see below)
1 in all the other cases

The **read_char** and **send_char** functions are the basic functions through which you can receive or send 8-bit data through the SCI. When using the **read_char** and the **send_char** functions, operations like interrupt handling for reception and transmission, reception buffer management and the XON-XOFF handshake (when set) are all done in the background, in a transparent way for your application.

The **cmd_interpreter** offers a complete solution for communication with the PC monitor - **MONPC28X**. By simply calling **cmd_interpreter** from your program, you can use **MONPC28X** commands to test and debug a program or save data during the normal operation of the program.

Normally when **cmd_interpreter** is called, it executes only a limited number of operations before passing back the control to the caller. In this way, **cmd_interpreter** avoids keeping too long the program control and increasing the processor overhead. The **cmd_interpreter** also passes back the control each time there is a dead time like waiting the next character to be received, or waiting the transmission to be reactivated (by a XON character), in order to send the data. This operation mode requires a periodic call of the **cmd_interpreter**. The recommended way is to include the call in an infinite loop, which always exists in a motion control program, usually in the main loop of the program (no interrupt loop).

Note:	The cmd_interpreter can be set with MONPC28X command p to use the checksum & acknowledge protocol (see par. 4.3.4.) which detects and eliminates possible errors during data exchanges with the PC.
-------	--

All the communication functions of **MON28335** respect the C calling conventions. Thus all the above functions can be called either from a program written in C, either from a program written in the assembly language.

4.4.4 Calling MON28335 Functions from C code

To call **MON28335** functions from a C code program, you have to:

- declare in their program a pointer to a function of the same type as **read_char**, **send_char** or **cmd_interpreter**
- set a pointer value equal with the **MON28335** function address
- call the function through this pointer

Example 4.12: calling **cmd_interpreter** from a C code

```
/* C header */
#define cmd_interpreter_addr 0x00330000 /* function address */
/* define a pointer to a function returning an int with void parameters and
assign its value to 0x00330000 */
int (*callmon28335)(void)=(int(*)())cmd_interpreter_addr;
```

```
/*----- Main code -----*/
void main(void)
{
    int stop;
    stop = 0;
    /*... your code ...*/
    while (!stop)
    {
        /*... your code ...*/
        ret_val_mon = (*callmon28335)(); /*Call cmd_interpreter*/
    }
}
```

4.4.5 Calling MON28335 Functions from assembly code

The **MON28335** functions can be called from assembly language by simply executing a call to their address.

Example 4.13: calling **cmd_interpreter** from ASM code

```
; Constant defines
MON28335      .set    0x00330000 ; command interpreter address

      .text
...
      /*... your code ...*/
...

      LCR      #MON28335      ; call command interpreter in a loop
...
```

Important: **MON28335** functions are written to be C callable. When any of these functions is called from assembly code, be sure that you respect C conventions for the caller.

4.4.6 Extending the command interpreter

Normally, when the **cmd_interpreter** is called from a program, it handles in a transparent way all the data reception and transmission. There is however an exception. This is when the **cmd_interpreter** recognizes a “user data” command. The code of this command is ‘**u**’. This means that data following this command code is not supposed to be read and interpreted by **cmd_interpreter**. This type of data is something that your program must read, analyze and interpret. For this reason, after a user data command, the **cmd_interpreter** does nothing but exits with return value **0**. This is the only situation in which the **cmd_interpreter** doesn’t return a **1**. Thus, the **cmd_interpreter** signals to the caller that he had received some data and it’s the caller responsibility to handle them. The **cmd_interpreter** simply expects to be called again after all the user data are received.

On its turn, the caller who expects some special data from time to time, must test the **cmd_interpreter** return value. If this is **0**, the caller will temporary disable the **cmd_interpreter** call from the loop and using **read_char** function will read his special data from the reception buffer. After the “end of transmission” character is received (this character is at your choice) the **cmd_interpreter** will re-enable the **cmd_interpreter** call.

This procedure allows you to extend the **cmd_interpreter** by constructing a 2nd interpreter for “user data”.

Lets take an example. Suppose you want to control when to execute a routine from your program, which sends 'q', each time is called. For this purpose you create two new commands: 'g' -go - enables the execution of this routine and 'k' - kill - disables its execution.

At the PC level, you need to press 'u' in order to set the monitor in terminal mode, then to press the letters 'g' or 'k'. To exit from terminal mode press the **ENTER**. This translates into the following operations:

- After 'u' is pressed **MONPC28X** switches to the **terminal operating mode**. It transmits to the MSK28335 DSC board the "user data" command code 'U'
- **MONPC28X** then sends the ASCII code of 'g', respectively 'k';
- finally the ASCII code of **ENTER** is send, which also ends the terminal mode and activates the normal operating mode.

At the DSC level, the following events occur:

- the command interpreter will read the 'u', and will return a 0;
- your program will read the 0 returned by the **cmd_interpreter** and will temporary disable further calls to **cmd_interpreter**;
- then calling the **read_char** function, your program starts reading the data. In this case the first character read is the 'g' or 'k' ASCII code;
- the 'g' or 'k' received character is used to select the activation or deactivation of the special routine;
- your program reads the **ENTER** code which in this example means: "end of user data" and re-enables the **cmd_interpreter** calling.

The corresponding DSC code can be:

Example 4.14:

```
/* define monitor, readchar and sendchar addresses */

#define cmd_interpreter_addr    0x00330000    /* address of cmd_interpreter*/
#define read_char_addr         0x00330002    /* for MON28335 v1.0.1 */
#define send_char_addr         0x00330004    /* for MON28335 v1.0.1 */

/* define pointers to the called functions */
int (*callmon28335)(void) = (int(*) (void))cmd_interpreter_addr;
int (*callreadchar)(void) = (int(*) (void))read_char_addr;
int (*callsendchar)(int) = (int(*) (int))send_char_addr;

void my_func(void)
{
    (*callsendchar)('q');    /* when called my_func sends 'q' */
}

void main(void)
{
    int ret_val_mon, stop, run_my_func, my_cmd;
    stop = 0;
    run_my_func = 0;
    ret_val_mon = 1;    /* must be initialized */

    while (!stop)
    {
        if (ret_val_mon)
            ret_val_mon = (*callmon28335)();
        else
    }
```

```
        {  
            my_cmd = (*callreadchar)();  
            switch (my_cmd)  
            {  
                case 'g': run_my_func = 1; break;  
                case 'k': run_my_func = 0; break;  
                case 0xd: run_my_func = 0; ret_val_mon = 1; break;  
            }  
            if (run_my_func && my_cmd != 0x200) my_func();  
        }  
    }  
}
```

4.4.7 Automatic execution of a program located into the flash at a specified address

The **ExtCodeFlash** demo is an application which starts to execute from TMS320F28335 flash automatically after reset or power on while still using the monitor for communication.

Lets consider an example. Suppose that we have a simple program written in C that calls the command interpreter in a loop (to have communication with the PC monitor) and in the loop we increment a variable just to be sure that we really execute this application. In order to make this application to start automatically after reset, the following steps must be followed:

- 1) Locate your application to any flash sector(s) except sector B (0x328000 - 0x32FFFF) which contains the monitor.
- 2) Change flash entry point locations, to branch after reset to your main application. This means to change the .cmd file, which should become:

MEMORY

```
{  
  
PAGE 0:  
    FLASHC      : origin = 0x328000, length = 0x008000  
    BEGIN       : origin = 0x33FFF6, length = 0x000002  
    ...  
PAGE 1:  
    FLASHDDM    : origin = 0x320000, length = 0x008000
```

SECTIONS

```
{  
    c_start:      {}          > BEGINPAGE 0  
    .text:        {}          > FLASHC      PAGE 0  
    .cinit:       {}          > FLASHC      PAGE 0  
    .pinit:       {}          > FLASHC      PAGE 0  
    .switch:      {}          > FLASHC      PAGE 0  
    IQmath:       {}          > FLASHC      PAGE 0  
  
    .const:       {}          > FLASHDDM    PAGE 1  
    .econst:      {}          > FLASHDDM    PAGE 1  
    .data:        {}          > FLASHDDM    PAGE 1  
  
    ...
```

- 3) Since after reset the TMS320F28335 jumps directly to your main function. This code, written in assembly language, should be included in a separate program section, which in the linker command file must be set to start at 0x33FFF6. For example, the settings can be:

```
.ref _c_int00
.sect "c_start" ; section with first instruction executed
.global _START
_START:
    LB _c_int00                ;Branch to start of boot.asm in RTS library
```

- 4) in order to use the monitor you need to start your application by performing the same initialisations the **MON28335** does. This operation means to call 4 initialisation routines of the **MON28335**.

Example 4.15:

```
#include "..\..\DSP28335_Regs.h"
#include "..\..\DSP28335_RegDef.h"
#include "mon28335.h"

int UserVar;
int ret_val_mon, stop;

void main(void)
{
    EALLOW;                // Enable writing to EALLOW protected registers
    SysCtrlRegs.WDCR= 0x0068;        // Disable watchdog module

    // reset watchdog counter
    SysCtrlRegs.WDKEY= 0x0055;
    SysCtrlRegs.WDKEY= 0x00AA;
    EDIS;                // Disable writing to EALLOW protected registers

    DINT; // Disable and clear all CPU interrupts:
    IER = 0x0000;
    IFR = 0x0000;

    /*the initialization routines must be called for establishing the
    communication with the board when the user's extended code is
    located into the flash*/
    (*CallInitializeInterrupts)();
    (*CallInitializeSystem)();
    (*CallInitializeData)();
    (*CallInitializeCOM)(B9600);

    EINT;                // Enable Global interrupt INTM
    ERTM;                // Enable Global realtime interrupt DBGM

    stop = 0;
    ret_val_mon = 1;                // must be initialized
    UserVar = 0;
    SetReady();                // Light on the led
```

```
        /*main loop*/
        while (!stop)
        {
            /*put your code here*/
            UserVar++;
            ret_val_mon = (*callmon28335)(); // call the monitor
        }
    }

/*****
// Turn on the led
*****/
void SetReady(void)
{
    EALLOW;
    GpioCtrlRegs.GPBQSEL2.bit.GPIO63 = 3;    // Asynch input GPIO63
    GpioCtrlRegs.GPBMUX2.bit.GPIO63 = 0;    // Select GPIO63 function

    GpioCtrlRegs.GPBDIR.bit.GPIO63 = 1;    // GPIO63 = output
    EDIS;

    GpioDataRegs.GPBCLEAR.bit.GPIO63 = 1;    // Turn on the light LED
}
```

4.4.8 RAM Memory restrictions

MON28335 uses locations **3F8000h - 3F807Fh** (from block **H0**) and **8000h – 8300h** (from block **L0**) of the 'F28335 DSC controller to store its internal variables. Your DSC program should not change these locations.

The **MSK28335** comes with 256K of external program/data memory, mapped from **0x00200000** to **0x0023FFFF**. This memory is free for use.

4.5 Flash Programming Tool

All the **MSK28335** and **MCK28335 kits** include a Flash programming tool through which you can to program the flash of DSC.

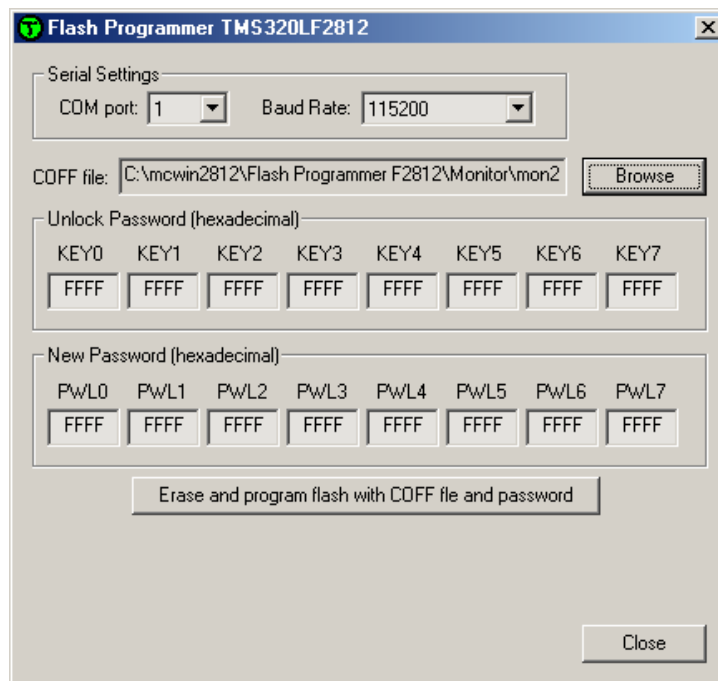


Figure 4.1. Flash Programmer 'F28335 window

Step to follow to program an **.out** file (**COFF**) into the flash memory:

- Select the SCI boot mode. See the table below:

JP1	OPEN
JP2	OPEN
JP3	OPEN
JP4	OPEN

- Connect the serial cable between **PC** and **MSK28335** board and the power supply
- Select the serial **COM port**
- Select the **Baud Rate**
- Select your **COFF file** (.out file with DSC program located in flash)
- Set the **Unlock Password**. By default, the device is unsecured and you must set
FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF

If the DSC was previously secured you must enter the password used previously for securing.

- Set the **New Password** for securing, which will be programmed into the flash (**PWL** registers). If the password is different than FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF (which leave DSC unsecured), you must note the password in order to reprogram the flash.

**CAUTION!**

DO NOT USE 128 BITS OF ALL ZEROS AS THE PASSWORD. THIS WILL AUTOMATICALLY SECURE THE DSC, REGARDLESS OF THE CONTENTS OF THE KEY REGISTER. THE DSC WILL NOT BE DEBUGGABLE NOR REPROGRAMMABLE.

DO NOT PULL A RESET AFTER CLEARING THE FLASH ARRAY BUT BEFORE ERASING THE ARRAY. THIS WILL LEAVE ZEROS IN THE PWL THAT WILL AUTOMATICALLY SECURE THE DEVICE, REGARDLESS OF THE CONTENTS OF THE KEY REGISTER. THE DEVICE WILL NOT BE DEBUGGABLE NOR REPROGRAMMABLE.

In order to reprogram the MON28335 monitor, program to flash the following COFF file:

<MxWIN28335 folder>\Flash Programmer F28335\Monitor\mon28335.out.

In order to execute the flash program the Flash boot mode must be selected with the following jumper configuration:

JP1	SHORT
JP2	SHORT
JP3	SHORT
JP4	SHORT

4.6 Creating a Customized PC Monitor using MONWIN32.DLL

All the **MSK28335** and **MCK28335** kits include the **MONWIN32.DLL** library - a collection of PC basic communication functions. These functions translate high level "monitor-like" commands into a set of elementary commands, which are recognized and executed by **MON28335** command interpreter at the DSC level.

With **MONWIN32.DLL** high level functions, you can easily create a customized version of a PC monitor, without knowing in details the PC-**MSK28335** communication protocol. **MONWIN32.DLL** functions can be used in any C or Visual Basic, 32 bits applications, written for Windows 95 / 98 / Me / NT or XP.

The **MONWIN32.DLL** library includes the following files:

- **MONWIN32.DLL** - dynamic link library containing all the communication functions
- **MONWIN32.LIB** - to be included in your application project
- **MONWIN32.H** - the header with functions prototypes for programs written in C
- **MONWIN32.BAS** - the header with functions prototypes for programs written in Visual Basic

4.6.1 MONWIN32.DLL User Callable Functions

The **MONWIN32.DLL** library contains functions organized in hierarchical levels.

At the lowest level are the elementary communication routines like: serial initialization, character transmission and reception.

At the highest level are the functions that implement the commands recognized by the **MON28335 cmd_interpreter**.

You have access to all the highest level functions plus the low-level functions mentioned above. Below is the list of the **MONWIN32.dll** library functions:

get_char	Gets the oldest character/byte received on the PC in the reception buffer, from the DSC board.
send_char	Transmits to the DSC board one character/byte, through the serial port of the PC.
set_address	Sends the Set current PM/DM address command to the MON28335 command interpreter. The next read / write monitor command will execute starting from the current PM / DM address.
send_data	Sends Write to PM/DM command to the MON28335 command interpreter. A block of data sent from the PC is written in the MCK28335/MSK28335 PM/DM memory.
get_data	Sends the Read from PM/DM command to the MON28335 command interpreter. A block of data is read from the MCK28335/MSK28335 PM/DM memory.
run	Sends Go to PM command to MON28335 command interpreter in order to start the execution of a previously downloaded DSC program. The DSC monitor executes a branch to the PM address.
set_baud_rate	Sends Set baud rate command to the MON28335 command interpreter and modifies the PC baud rate for the serial port used for the communication.
xonxoff	Sends the Enable/Disable XON-XOFF handshake command to the MON28335 command interpreter.
check_ack	Sends the Enable/Disable MCK28335 checksum and acknowledge protocol command to the MON28335 command interpreter.
reset	Reset the DSC
download_RAM	Download a COFF file to RAM memory

The description of these functions (calling conventions, input / output arguments) is given below.

int WINAPI get_char(BYTE* data);

Gets the oldest character/byte received on the PC in the reception buffer, from the DSC board.

Input arguments:

data: pointer to a buffer that receives the character

Returns:

FALSE: no character received in the last time out period (time-out error).

TRUE: reception OK. The **data** points to the received character/byte.

int WINAPI send_char(BYTE data);

Transmits to the DSC board one character/byte, through the serial port of the PC.

Input arguments:

Data: data to be sent

Returns:

TRUE: if the character is sent.

FALSE: if the character isn't sent because of transmission error.

int set_address (unsigned long addr, int mem_type)

Sends the *Set current PM/DM address* command to the **MON28335** command interpreter. The next read / write monitor command will execute starting from the current PM / DM address.

Input arguments:

addr : new current PM/DM address
mem_type: memory type: **0** : program memory (PM), **1** : data memory (DM)

Returns:

1 : if successful
0 : if not successful

Note:	The current PM/DM addresses must be kept synchronized at the DSC level and at the PC level. For this reason the set current PM/DM address command must be executed prior to any other command that uses the current PM/DM address. After a set current PM/DM address command, both at the DSC level and at the PC level the variables that keep track of current PM/DM address take the same value. Any other command that leads to the modification of the current PM/DM address at DSC level updates also the counterpart variables at the PC level in order to keep the synchronisation.
-------	---

int WINAPI send_data(WORD * data, int mem_type, WORD data_len);

Sends *Write to PM/DM* command to the **MON28335** command interpreter. A block of data sent from the PC is written in the PM/DM memory.

Input arguments:

data: pointer to the data block to be sent. 1st sent is data[0], then data[1], etc.
mem_type: memory space where the data will be stored
data_len: number of words to be stored

Returns:

TRUE: if successful
FALSE: if not successful

Notes:	1). If the command is successful, the new current PM/DM address (both at PC level and at DSC level) is incremented with data_len + 1 , i.e. with the data block length. This means that after a prior setting of the current PM/DM address, as long as data are to be written at consecutive addresses in the PM/DM memory space, no other current address setting is necessary. The next set PM/DM current address command will be sent only when data are to be written starting from an address that differs from the updated current address.
	2). Before sending the data block, this command checks if the length of the data block to be sent is the same with the current PM/DM data block length. If these lengths are not equal, first a set PM/DM data block length command is executed in order to set the PM/DM data block length equal with data_len , and only then the data block is sent. This approach make transparent the data block length management that is automatically adjusted depending on the length of data block to be sent. The only requirement is to perform an initial synchronization. Sending only once a set PM/DM data block length command, before starting the data exchanges does this.
	3). In order to keep reasonable the reception buffer dimensions at the DSC level, the maximum data block length that can be sent once is 8 words (maximum data_len = 7)

int get_data (unsigned int *data, int mem_type, unsigned char data_len);

```
int WINAPI get_data( WORD* data, int mem_type, WORD data_len );
```

Sends the **Read from PM/DM** command to the **MON28335** command interpreter. A block of data is read from the IMMC24x PM/DM memory and sent to the PC. After each word sent to the PC.

Input arguments:

data: a pointer to the received data block. The 1st received is data[0], then data[1], etc.
mem_type: memory space from where the data will be stored
data_len: number of words to be stored

Returns:

TRUE: if successful
FALSE: if not successful

Note:	All the remarks made for send_data also apply to the get_data function.
-------	---

```
int WINAPI run(void);
```

Sends **Go to PM** command to **MON28335** command interpreter in order to start the execution of a previously downloaded DSC program. The DSC monitor executes a branch to the entry point PM address.

Input arguments:

-

Returns:

TRUE: if successful
FALSE: if not successful

```
int WINAPI set_baud_rate(BYTE baud);
```

Sends **Set baud rate** command to the **MON28335** command interpreter and modifies the PC baud rate for the serial port.

Input arguments:

nBaud: new baud rate; only B9600, B19200, B38400, B56000 and B115200 values are accepted. See "Monwin32.h" and "windows.h" header files for more details.

Returns:

TRUE: if successful
FALSE: if not successful

Note:	The default baud rate is 9600 . Before closing the PC monitor the 9600 baud rate should be restored.
-------	---

```
int WINAPI xonxoff(int status);
```

Sends the **Enable/Disable XON-XOFF handshake** command to the **MON28335** command interpreter.

Input arguments:

Status: 1 enables the handshake , other value disables the handshake

Returns:

TRUE: successful
FALSE: fail

Note:	The default handshake status is disabled. Before closing the PC monitor the XON-XOFF
-------	--

status should be disabled.

int check_ack (int status)

Sends the *Enable/Disable checksum and acknowledge protocol* command to the **MON28335** command interpreter.

Input arguments:

status : 1 enables the protocol, other value disables the protocol

Returns:

1 : if successful
0 : if not successful

Note:	The default protocol status is disabled. Before closing the PC monitor, the checksum & acknowledge protocol should be disabled.
-------	---

int WINAPI reset (void)

Sends "Reset" command to MON28x command interpreter in order to reset the DSC.

Input arguments:

-

Returns:

1 : if successful
0 : if not successful

int WINAPI download_RAM (int mode, const char *file_name, int (stdcall *pfCallbackFunction)(int, const char*, void*), void* usrData, DWORD* p_entry)

Downloads a COFF object program to MSK28335 board in RAM memory.

Input arguments:

mode: 0, performs a 'quiet' download. After download program is automatically executed. No message returned through the 'callback' function.
 1, performs a 'verbose' download. After download, an OK message can be received through callback function. Program is not executed. A separate run command must be sent by the user.

file_name: provides the COFF file name to download;

pfCallbackFunction pointer to callback function:
 0, the 'callback' function is not called;
 not 0, the 'callback' function is called; A valid data must be provided.

usrData pointer to data passed to callback function;

p_entry "entry point" variable:
 0, no entry point returned;
 not 0, the COFF entry point will be returned in this variable; A valid data must be provided.

Returns:

1 : RAM download is successful;
0 : an error occurs while reading file to download: file not found or file not opened correctly or file has wrong format (is not COFF);
2 : a communication error occurs during PC - MSK28335 data exchange;

4.6.2 Guidelines for MONWIN32.DLL Functions Use

In order to correctly use the **MONWIN32.DLL** functions, the following guidelines should be considered:

- check first if the PC serial port intended to be used is available;
- call **set_address** and **set_length** to synchronize the internal variables which keep track of the current PM/DM address, respective the current PM/DM data block length at the PC and at the DSC level;
- don't forget to set the current PM/DM address before a **send_data** or **get_data** call. If data are to be written to, or read from consecutive addresses in the **MSK28335 DSC board** memory, set only once the current PM/DM address, before the first **send_data** or **get_data** call;
- before closing your application set the default settings of the PC - **MSK28335 DSC board** communication: 9600 baud, no XON-XOFF handshake, no checksum & acknowledge protocol;

5 Processor Evaluation Applications

This chapter describes the structure and functions of the **PROCEV2833x** program for the evaluation of the 'F28335 DSC controller.

Contents

- 5.1. Processor Evaluation Concept**
- 5.2. Analog to Digital Converter Module Application**
- 5.3. Capture Units Application**
- 5.4. Digital I/O Ports Application**
- 5.5. PWM Wave Form Generation Application**
- 5.6. Quadrature Encoder Pulse (QEP) Circuit Application**
- 5.7. Serial Communication Interface (SCI) Module Application**
- 5.8. Serial Peripheral Interface (SPI) Module Application**
- 5.9. Enhanced Controller Area Network (eCAN) Module Application**
- 5.10. General Purpose Timer (GPT) Application**
- 5.11. Watchdog (WD) Module Application**
- 5.12. Inter-Integrated Circuit (I2C) Module Application**
- 5.13. Multichannel Buffered Serial Port (McBSP) Module Application**

5.1 Processor Evaluation Concept

The **PROCEV2833x** program offers several simple ready-to-run applications on the **MSK28335 DSC board**. You can program and test the basic I/O functions of the DSC chip (as timers, PWM, captures, QEP, GPIO, SCI, SPI, CAN, watchdog, A/D converters).

The main purpose of this program is to offer, on one side, at Windows level, a simple to use, effective tool for the 'F28335 DSC controller I/O evaluation. On the other side, the program allows you to examine the source code of all the included DSC examples. Thus, **PROCEV2833x** provides the basic I/O functions needed to implement a motion application. These functions can be partly or completely included in your application.

Note that the **PROCEV2833x** application use DSC demos from the folder **<MxWIN28335 folder>\Demos**.

5.1.1 Invoking the program

The program can be activated from the control panel platforms by clicking on the «**Processor Evaluation**» icon.

Once the program is activated, the **Processor Evaluation Control Panel** window (Figure 5.1) appears on the screen.

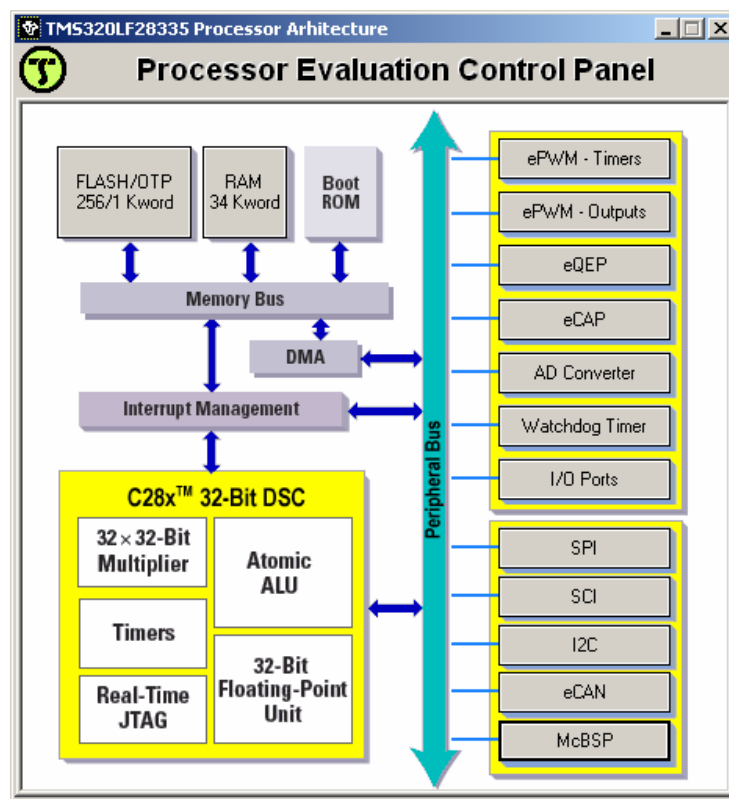


Figure 5.1. Processor Evaluation Control Panel window

The following applications can be selected in this control panel:

AD Converter	Analog to Digital Converter Module Application - evaluates A/D conversion possibilities offered by the on-chip Analog to Digital Converter Module
eCAP	Capture Units Application - experiments event capture features on any of the 3 existing capture units of the Event Manager A.
I/O Ports	Digital I/O Ports Application - tests General Purpose Input / Output Port B of the processor, for user defined configurations of individual I/O bits.
ePWM - Outputs	PWM Waveform Generation Application - experiments different possibilities to generate PWM waveforms with the on-chip PWM module.
eQEP	Quadrature Encoder Pulse (QEP) Circuit Application - obtains position information from an incremental encoder using the Event Manager A's QEP Circuit.
SCI	Serial Communication Interface (SCI) Module Application - programs and controls a serial communication between the PC and the MSK28335 board.
SPI	Serial Peripheral Interface (SPI) Module Application - sends/receives user specified character codes to/from the high speed on-chip Serial Peripheral Interface.
eCAN	Enhanced Controller Area Network (eCAN) Module Application - sends/receives a user specified CAN message.
ePWM - Timers	General Purpose Timer (GPT) Application - evaluates some of the features offered by the ePWM Timers.
Watchdog Timer	Watchdog (WD) Module Application - experiences the features offered by the Watchdog Module.
I2C	Inter-Integrated Circuit (I2C) Module Application - sends/receives user specified character codes to/from the high speed on-chip I2C Interface.
McBSP	Multichannel Buffered Serial Port (McBSP) Module Application - sends/receives user specified character codes to/from the high speed on-chip McBSP Interface.

5.1.2 Applications' features

The **PROCEV2833x** module consists on two separate components: the Windows program on the PC side, and the different DSC applications on the **MSK28335 DSC board** side.

The Windows part of the program, from the PC, communicates with the DSC board and uses (transparently for you) the **MONWIN32.DLL** communication module functions (see chapter 4 for details about **MONWIN32.DLL** module). Thus it is possible that the PC program examines and/or modifies the contents of specific data memory cells of the DSC board.

The DSC applications are structured in such a manner that they are self-contained and independent of the PC program. The only element that was used in order to allow the communication between the PC and the **MSK28335 DSC board**, during the execution of the DSC application examples, is a link to the monitor of the DSC board. Thus, almost each of the DSC application examples contains a call to the monitor command interpreter function resident in the flash memory of the DSC board (see chapter 4 for details about the **MON28335** monitor program). This allows the PC program to dynamically modify and/or read the contents of specific variables of the application examples, from the data memory of the **MSK28335 DSC board**, during the execution of these examples.

Figure 5.2 presents the basic concept of the data structure and flow between the PC and **MSK28335 DSC board**. As one can see, different variables, which are set in the Windows environment on the PC computer, are loaded through the communication channel into the data memory of **MSK28335 DSC board**.

When you press the “Run” button, the DSC application is downloaded on the **MSK28335 DSC board**, the application specific variables are initialized into the data memory of the DSC, corresponding to the PC window selections done by you, and the DSC application is started. As stated before, after an initialization sequence, the DSC application performs specific I/O functions associated to the tested chip function, and calls the command interpreter monitor function.

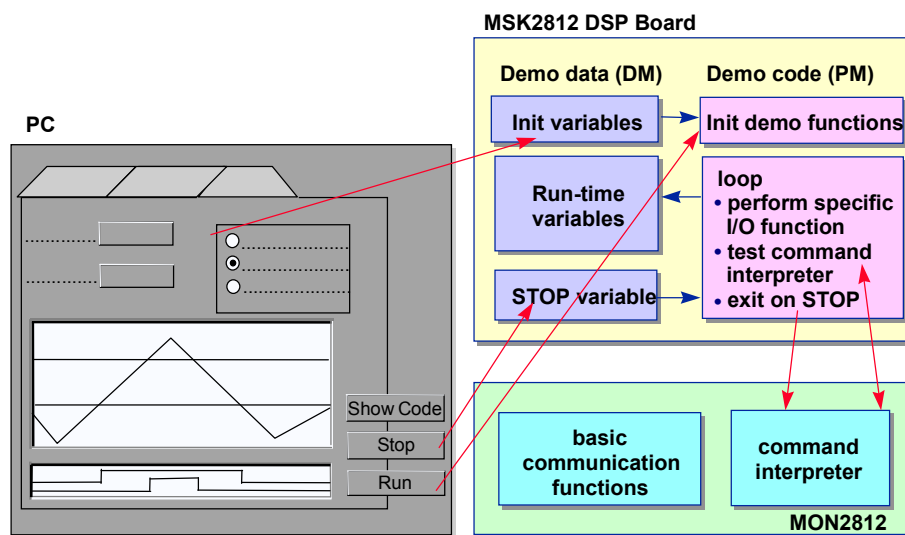


Figure 5.2. PROCEV2833x module - non-interrupt application demo structure

Also common to the majority of the DSC applications, a specific variable named “Stop” is tested in order to detect the end of the test. The PC program sets this variable when you press the “Stop” button in the dialog screen. At that moment, the DSC demo program is stopped and the control is returned on the **MSK28335 DSC board** to the **MON28335** communication monitor program.

Remark: Depending on the demo application, some of the parameters that may be defined in the Windows environment are loaded only at the start of the demo application. If you want to modify them, the DSC application must be stopped (using the “**Stop**” button”), the parameters may be modified, and the application may be started again (using the “**Run**” button). Other parameters may be modified on-line, during the execution of the demo, without the need to stop it. These specific aspects will be outlined for each of the demo applications, described latter in this chapter.

A similar communication scheme is implemented for the demo applications that use some interrupts on the **MSK28335 DSC board**. In these cases, the DSC program uses the specific implementation of the interrupt structure. Figure 5.3 presents the basic structure of the DSC board programs in this case.

In order to have a direct link between the Windows program and the DSC applications, the **PROCEV2833x** module uses specific names for the variables of the DSC code. These variable names are displayed for the initialization and settings parameters of the I/O functions, at Windows level. Thus, when setting specific parameters for a demo, you get the image of the corresponding binary or hexadecimal value that must be stored in a specific memory-mapped register of the DSC processor. By examining the associated source code of the application, you get a direct feeling of the parameters setting procedure and their specific values.

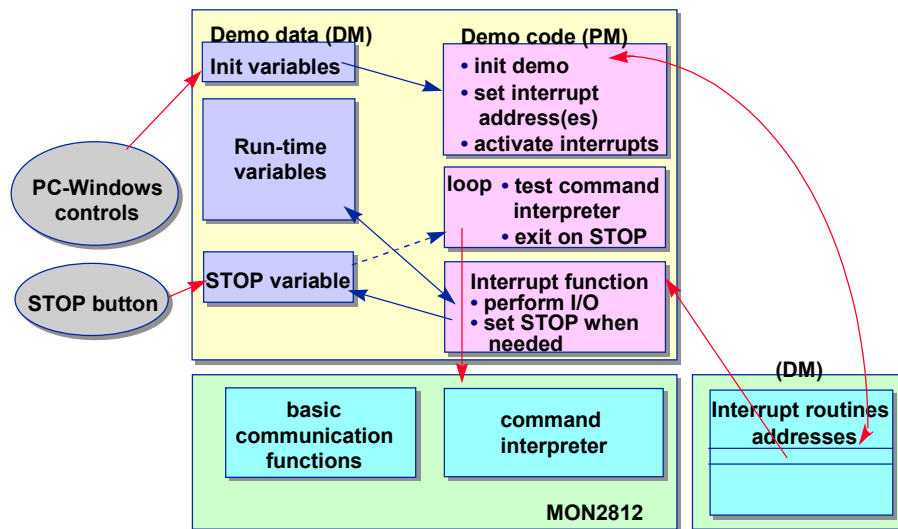


Figure 5.3. *PROCEV2833x module - interrupt-based application demo structure*

All the proposed DSC applications can also be loaded and executed directly to the **MSK28335 DSC board** from the **monitor** program. By launching the monitor communication, you can load the corresponding *.out object file on the DSC board. Then, based on the description of each application, you must set the appropriate values for the initialization variables - as described for each application in the following paragraphs. The program can then be executed (using the “**Run**” monitor command). Using monitor commands (as “**Inspect**”, “**Modify**”, “**Adjust**”), you can then examine/modify the corresponding application variables. In order to stop the DSC application execution, you must set (using the “**Modify**” monitor command) the “**Stop**” variable of the DSC code.

All Windows applications dialogs have **6 action control buttons** with common purpose in any application. By activating these buttons, you can start or stop the application, close the application dialog and return to

the Processor **Evaluation Control Panel**, examine the project files associated to the application, activate the PC Monitor (**MONPC28X**) program or get interactive on-screen help about the application description.

The **action control buttons** are situated on the up-right corner of every application dialog, their functionality being described below:



- start the application
- stop the application
- activate a panel to examine application project files and DSC code
- start **MONPC28X** program
- offer on-screen help about the application
- close the application and return to **Processor Evaluation Control Panel**

The following paragraphs describe in detail each application's purpose, screen presentations, actions to run the application, associated DSC program flowchart, project files and variables and application result evaluation.

5.2 Analog to Digital Converter Module Application

5.2.1 Purpose

Evaluates A/D conversion possibilities offered by the on-chip 12-bit analogue-to-digital converter (ADC) module. This peripheral has sixteen multiplexed analogue inputs and a fast conversion time of 60ns.

The ADC module consists of two independent 8-state sequencers (SEQ1 and SEQ2) that can be operated individually in “dual-sequencer mode” or cascaded into one large 16-state sequencer (SEQ) in “cascaded mode”. In both cases, the ADC has the ability to auto sequence a series of conversions. For every conversion, any one of the available 16 input channels can be selected through the analogue multiplexer. After conversion, the digital value of the selected channel is stored in the appropriate result register (RESULTn). It is also possible to sample the same channel multiple times, allowing the user to perform “over-sampling”, which gives increased resolution over traditional single sampled conversion results.

The application performs analogue-to-digital conversion of up to sixteen user-selected channels at a programmed acquisition rate. The timer of ePWM1 module is used to generate the start of conversion with the acquisition rate. The ADC interrupt is used for storing of acquisition results in a buffer of maximum 20480 words length.

5.2.2 Screen Presentation

The dialog presented in Figure 5.4 is accessible by selecting the « **A/D Converters** » button in the **Processor Evaluation Control Panel** menu.

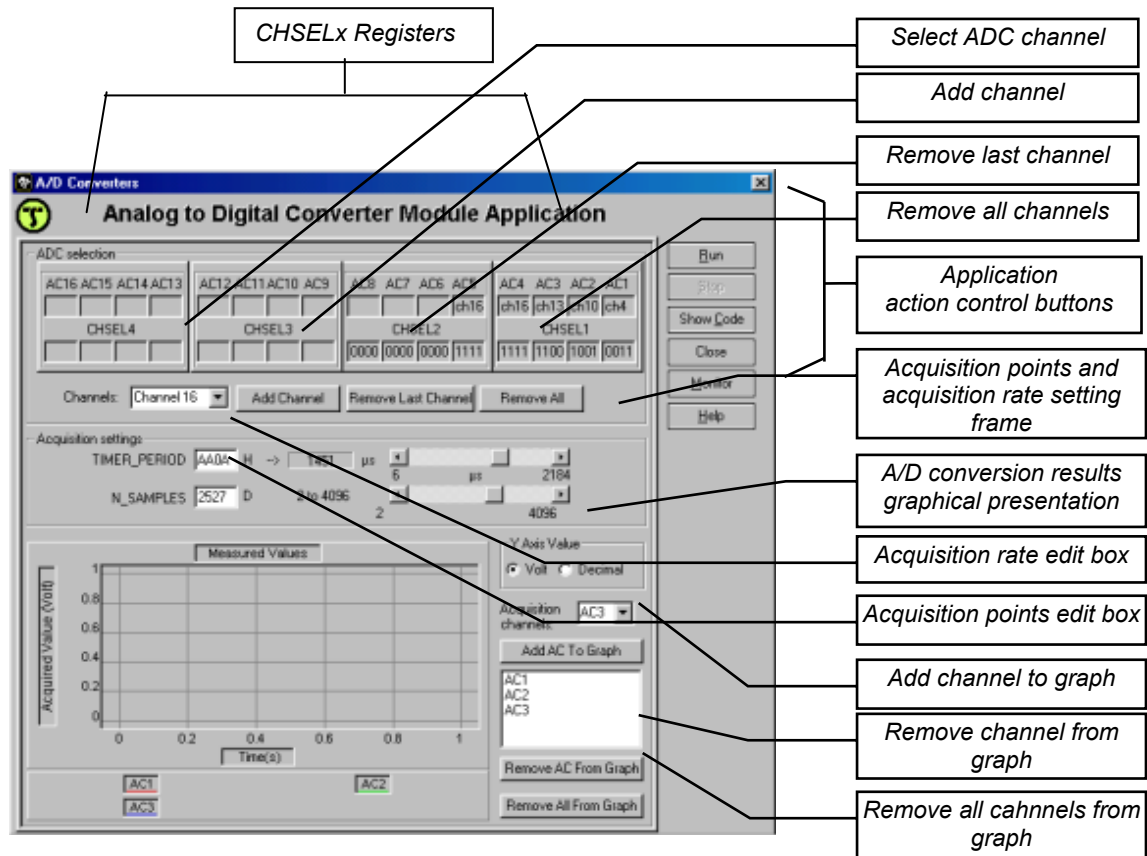


Figure 5.4. Analog to Digital Converter Module Application dialog

5.2.3 Actions to run the application

Step 1.

- In the **Channels** combo box, select the desired channel and add that using **Add Channel** button

Step 2.

- Set the timer period (**TIMER_PERIOD**) to define the acquisition rate. Use either the Acquisition rate edit box or the associated scroll bar.
- Set the number of acquisition points (**N_SAMPLES**). Use either the Acquisition points edit box or the associated scroll bar.

Step 3.

- At **Acquisition channels** select the channel which must be displayed (AC1, AC2, ...) and press **Add AC to To Graph** button. Just 4 channels can be simultaneously displayed.

Step 4.

- Click on the **Run** button to start the experiment. Wait for the A/D conversions to be done. If the selected acquisition rate is too slow or if the desired number of acquisition points is too big, the conversions can be stopped at any moment using the **Stop** button. Wait for uploading and plotting of the acquisition data.

5.2.4 DSC software description

Figure 5.5 presents the flowchart of the DSC program that performs this application.

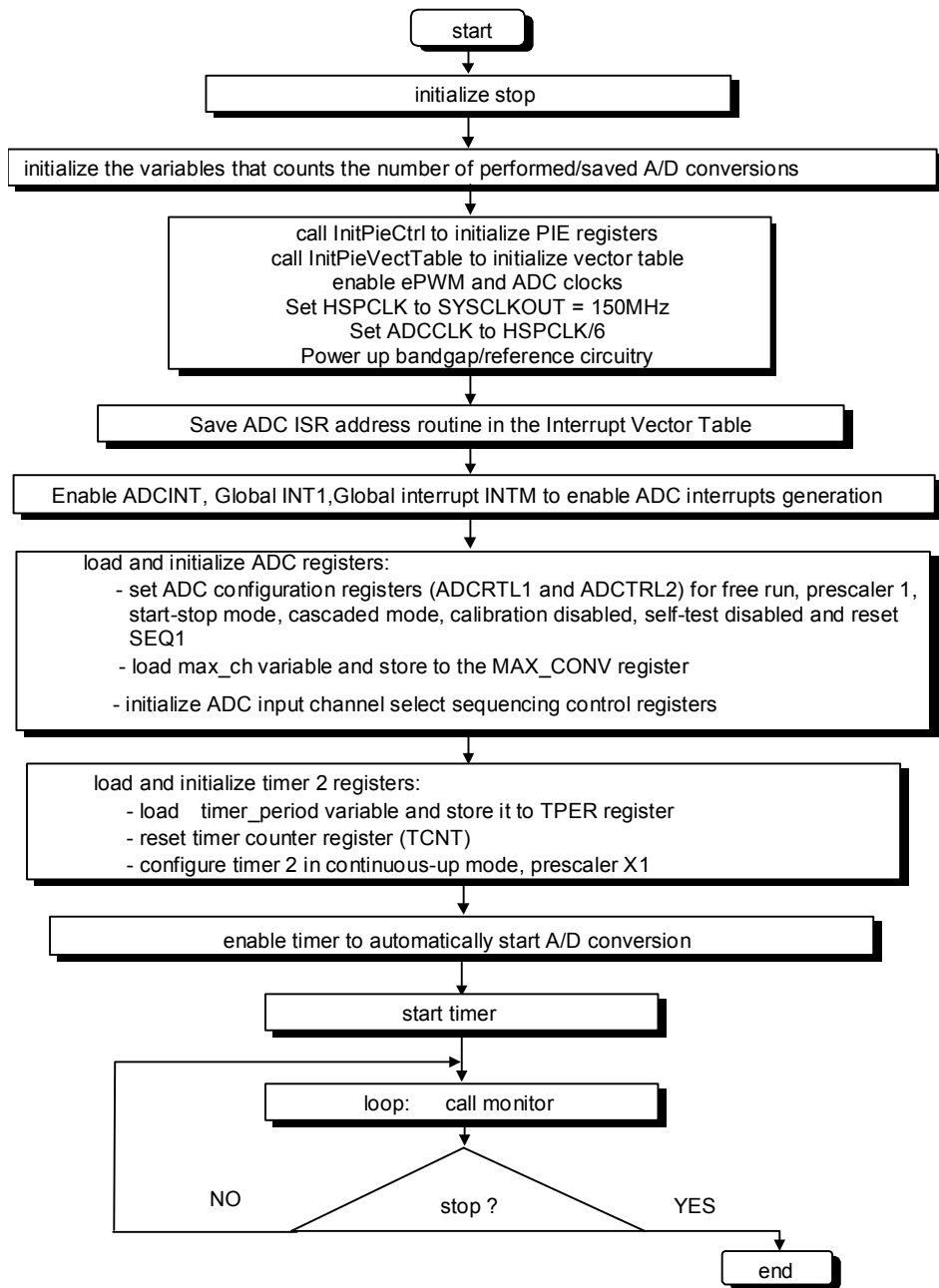


Figure 5.5.A. Dual Analog to Digital Converter Module Application DSC program flowchart

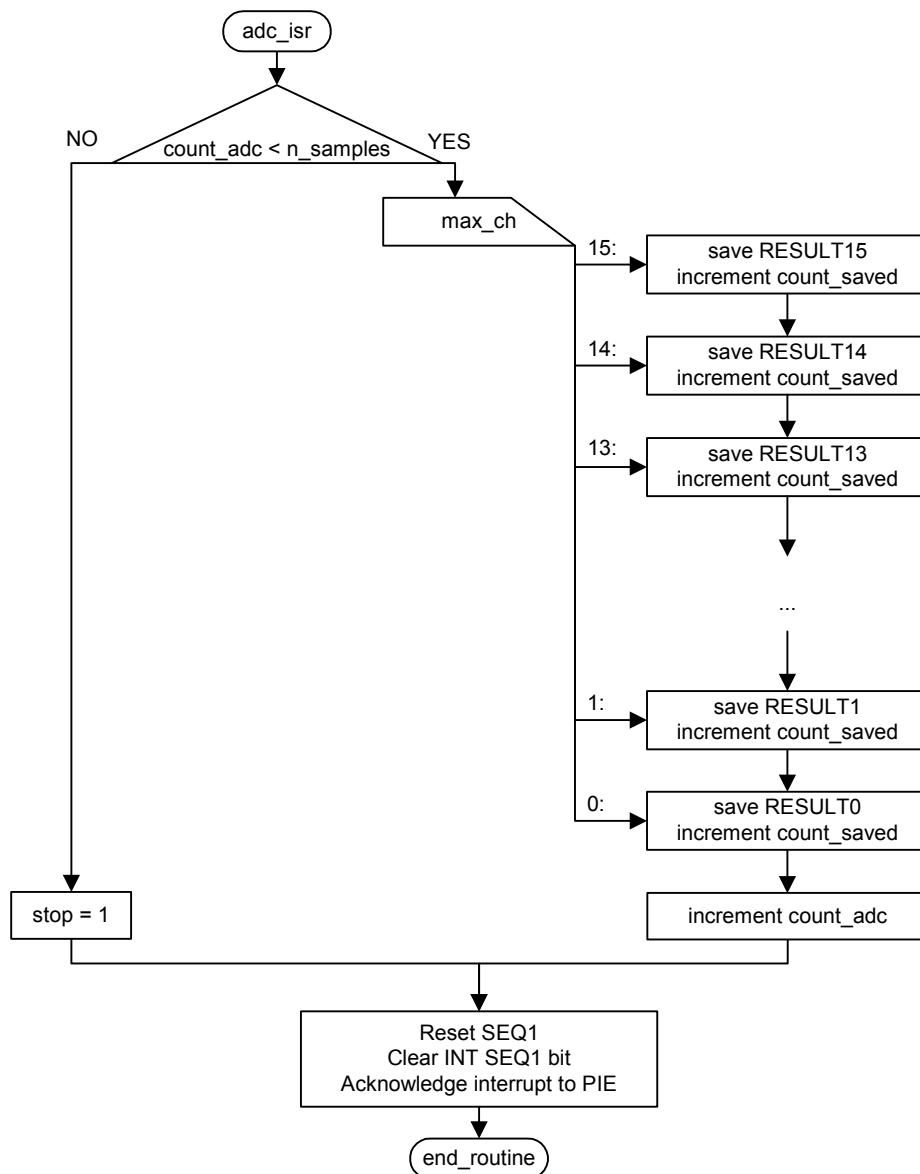


Figure 5.5.B. Dual Analog to Digital Converter Module Application DSC program flowchart

5.2.5 Program files

The file name containing the DSC program for this application is **adc.c**. Activating the Show code action control button in the application dialog can inspect it. The C header file called **adc.h** holds the variables and function declarations for the application.

5.2.6 Variables and functions

The program consists of a **main function** with the entry point at internal program memory address **0x3f8080**, and an **interrupt service routine** function: **adc_isr**.

Program variables are allocated in internal data memory starting at address **0x3f9000**:

0x3f9000: DemoADC.stop	- program stop index (set to 1 to end the program)
0x3f9001: DemoADC.ch_sel1	- variable containing the sequencing control register 1 (CHSELSEQ1)
0x3f9002: DemoADC.ch_sel2	- variable containing the sequencing control register 2 (CHSELSEQ2)
0x3f9003: DemoADC.ch_sel3	- variable containing the sequencing control register 3 (CHSELSEQ3)
0x3f9004: DemoADC.ch_sel4	- variable containing the sequencing control register 4 (CHSELSEQ4)
0x3f9005: DemoADC.max_ch	- variable containing the value for the maximum conversion channels register (MAX_CONV)
0x3f9006: DemoADC.timer_period	- variable containing the programmed acquisition rate, it is loaded to Timer Period Register T2PER
0x3f9007: DemoADC.n_samples	- variable containing the number of A/D conversions to be made
0x3f9008: DemoADC.count_adc	- variable containing the current number of A/D conversions made
0x3f9009: DemoADC.count_saved	- variable incremented each time an A/D conversion result is saved into the result buffer
0x3f900A: DemoADC.ret_val_mon	- value returned by the monitor
0x00C000: res_buf	- buffer of maximum 4096 words, containing the A/D conversions saved results

5.2.7 Application results evaluation

The application experiments the use of the ADC module existing on TMS320F28335 processor. Up to 16 analog input channels can be selected to measure voltages between 0 and 3 Volts. A signal generator in this voltage range can be connected to any of the A/D channels existing on **J3** and **J4** connectors of the MSK28335 board (ADCINA0 – ADCINA7 and ADCINB0 – ADCINB7 are available on the two specified connectors). Up to 4096 acquisition points per each channel can be performed when running this experiment. The acquisition rate can be programmed between 1 and 436 microseconds. The acquisition data can be evaluated (either in hexadecimal values or in volts) based on the graphical presentation provided in the application view.

5.3 Capture Units Application

5.3.1 Purpose

Experiments event capture features on each one of the 6 capture modules.

The application helps to understand how different external signal transitions (rising edge or falling edge) can be detected by the capture modules. External connections to the capture input pins are required to generate capture events. When a capture unit is enabled and the application is started, the specified transition on the associated input pin causes the counter value of the timer to be locked into the corresponding Capture Register (CAP1...4). The corresponding status bits in ECFLG register are adjusted and the application view reflects the new status of the captures (1 capture or 2 captures = **OVERflow**) each time a new counter value is captured into the Capture Register (CAP1...4).

5.3.2 Screen Presentation

The dialog shown in Figure 5.6 is accessible by selecting the « **Captures** » button in the **Processor Evaluation Control Panel** menu.

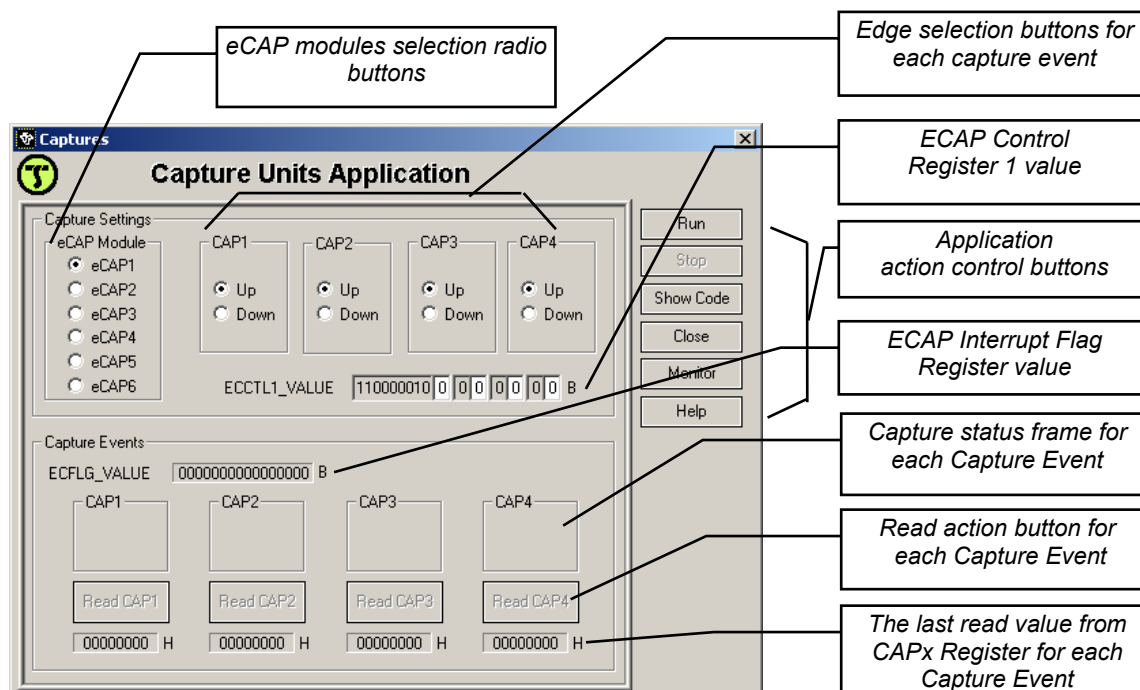


Figure 5.6. Capture Units Application dialog

5.3.3 Actions to run the application

Step 1. Click on the edge selection buttons to enable one or more capture units to be active during the application.

Step 2. Click on the **eCAP1...6** selection buttons to capture module.

Step 3. Click on the **Run** button to start the experiment. Use external connections to the enabled capture units input pins to generate edge transitions. When a specified transition happens, the counter value of the timer will be captured in the respective capture event register (CAP1...4). The event will be reported in the application view both in the capture status frame and in the corresponding bits of the ECAP Interrupt Flag Register (**ECFLG_VALUE**). Click on the **Read CAP1...4** action buttons to read the captured counter value. The respective hexadecimal value will be displayed in the associated text box.

5.3.4 DSC software description

Figure 5.7 presents the flowchart of the DSC program that performs this application.

5.3.5 Program files

The file name containing the DSC program for this application is **cap.c**. Activating the Show code action control button in the application dialog can inspect it. The C header file called **cap.h** holds the variables and function declarations for the application.

5.3.6 Variables and functions

The program consists of a main function with the entry point internal program memory address **0x3f8080**.

Program variables are allocated in external data memory starting at address **0x3f9000**:

0x3f9000: DemoCAP.stop	program stop index (set to 1 to end the program)
0x3f9001: DemoCAP.ecctl1_value	variable containing the value for ECAP Control Register 1 (ECCTL1)
0x3f9002: DemoCAP.status	variable holding the content of the Interrupt Flag Register (ECFLG)
0x3f9003: DemoCAP.read_cap_no	variable representing the number of the capture event whose Capture Register the user wants to read. Each time user clicks on a Read CAPx action button, the program that controls the application view on the PC, loads the corresponding capture event number into this variable.
0x3f9004: DemoCAP.cap1	variable holding the last value read from the Capture 1 Register (CAP1)
0x3f9006: DemoCAP.cap2	variable holding the last value read from the Capture 2 Register (CAP2)
0x3f9008: DemoCAP.cap3	variable holding the last value read from the Capture 3 Register (CAP3)
0x3f900A: DemoCAP.cap4	variable holding the last value read from the Capture 4 Register (CAP4)
0x3F900A: DemoPWMA.selected_module	variable specifying which capture module (eCAPx) are to be used

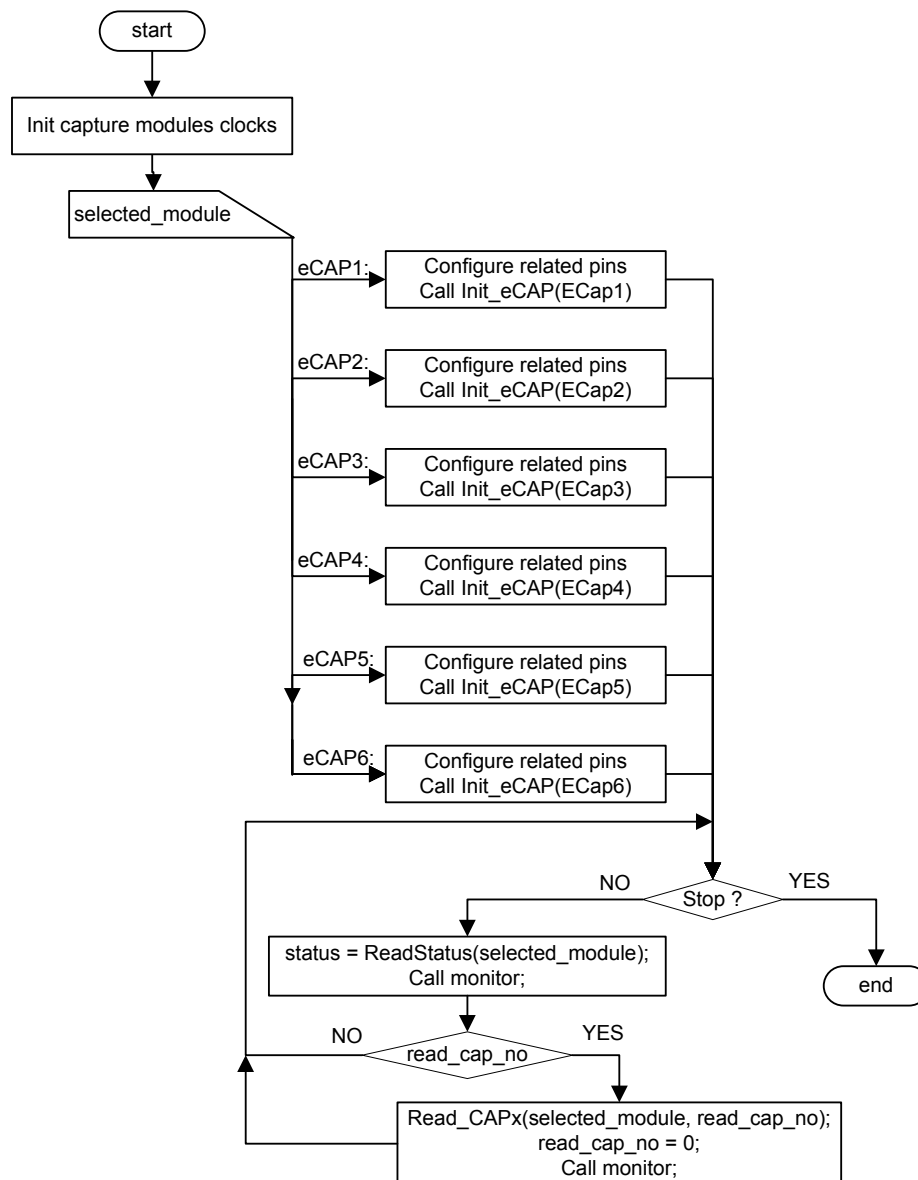


Figure 5.7.A. Capture Units Application DSC program flowchart

5.3.7 Application results evaluation

The application tests external events monitoring with the on-chip Capture Modules. Only one Capture Module can be enabled during the experiment. External connections are needed in order to generate transition on the Capture Modules input pins. These pins are available on **J3** and **J4** connector (pins 13 to 15) (see Appendix A for hardware specifications).

5.4 Digital I/O Ports Application

5.4.1 Purpose

Test Input / Output Ports - 16...27 of the processor, for user defined configurations of individual I/O bits. The application allows configuration of 12 individual pins as inputs or outputs. Logical values 0 or 1 can be sent to the outputs, while the inputs are continuously monitored and updated in the application dialog.

5.4.2 Screen Presentation

The dialog presented in Figure 5.8 is accessible by selecting the « I/O Ports » button in the **Processor Evaluation Control Panel** menu.

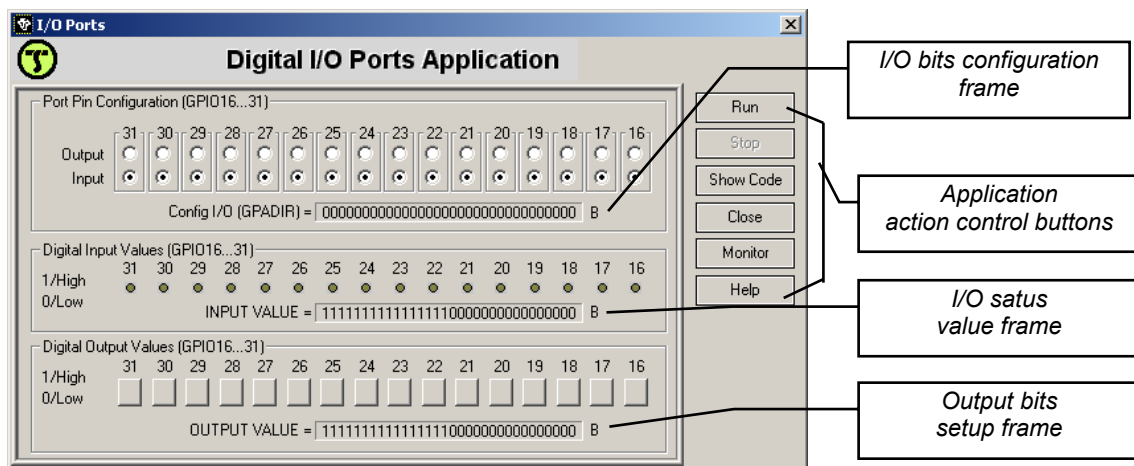


Figure 5.8. Digital I/O Ports Application dialog

5.4.3 Actions to run the application

Step 1. Use the mouse in the I/O bits configuration frame to select the individual bits as input or output as desired.

Step 2. In the Output bits value setup frame select the logical value 0 or 1 to be sent to each of the individual bits configured as outputs.

Step 3. Click the **Run** button to start the experiment.

Step 4.

- Measure the voltage on the I/O pins configured as outputs to check the similarity with the values selected at Step 2
or

- Choose one or several I/O pins configured as inputs at Step 1 and connect to ground to see the change of values in the I/O status value frame

Note: Processor GPIO16 – GPIO27 pins are accessible through the **J1, J2, J4** and **J10** connectors (see Appendix B for pins specifications).



CAUTION! DO NOT APPLY A POTENTIAL ON THE INDIVIDUAL PINS CONFIGURED AS OUTPUTS! PERMANENT DAMAGE OF THE DSC BOARD MAY BE PRODUCED IN THIS CASE!

5.4.4 DSC software description

Figure 5.9 presents the flowchart of the DSC program that performs this application.

5.4.5 Program files

The file name containing the DSC program for this application is **gpio.c**. Activating the Show code action control button in the application dialog can inspect it. The C header file called **gpio.h** holds the variables and function declarations for the application.

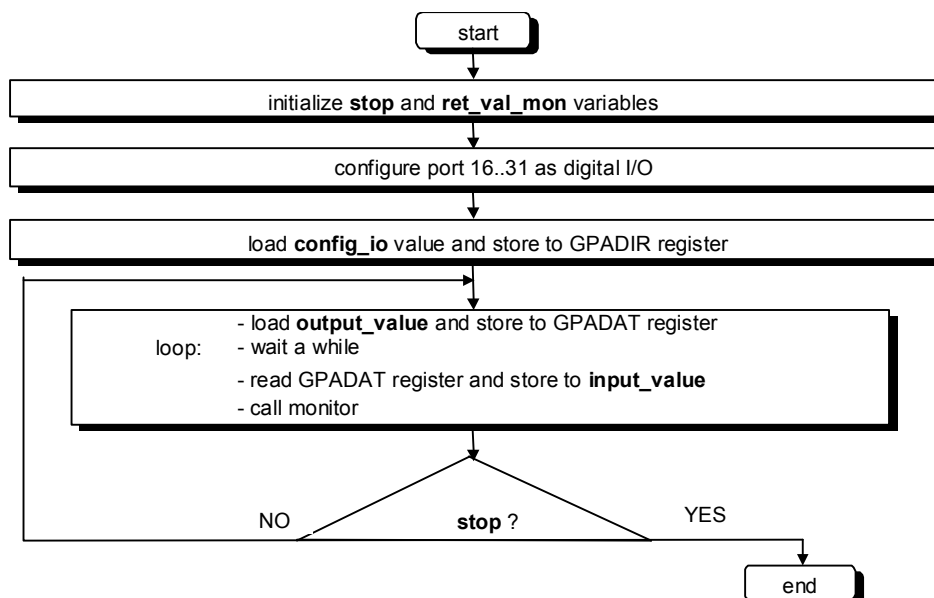


Figure 5.9. Digital I/O Ports Application DSC program flowchart

5.4.6 Variables and functions

The program consists of a **main function** with the entry point internal program memory address **0x3F8080**.

Program variables are allocated in external data memory starting at address **0x3F9000**:

0x3F9000: DemoGPIO.stop	program stop index (set to 1 to end the program)
0x3F9001: DemoGPIO.config_io	the configuration value of GPADIR register. Output (bit=1) or input (bit=0).
0x3F9003: DemoGPIO.input_value	the value that contains the 16 bit value read from ports
0x3F9005: DemoGPIO.output_value	the value that is written to ports 16...31 to determine the output pins to be in Low/High state

5.4.7 Application results evaluation

The logical value of an input pin (as well as the logical values of the output pins) can be monitored on-line in the I/O status value frame.

The logical value of each individual output pin can be modified on line in the Output bits value setup frame. The output values can be measured with a voltmeter or an oscilloscope on the specific pins of the **J1**, **J2**, **J4** and **J10** connectors.

5.5 PWM Waveform Generation Application

5.5.1 Asymmetric PWM

5.5.1.1 Purpose

Generate three-phase voltages using asymmetric pulse width modulation (PWM) technique.

ePWM modules 1...3 or 4...6 of controller are programmed to give 6 PWM signals in PWM asymmetric mode.

5.5.1.2 Screen Presentation

The dialog presented in Figure 5.10 is accessible by selecting the « **PWM** » button in the **Processor Evaluation Control Panel** menu.

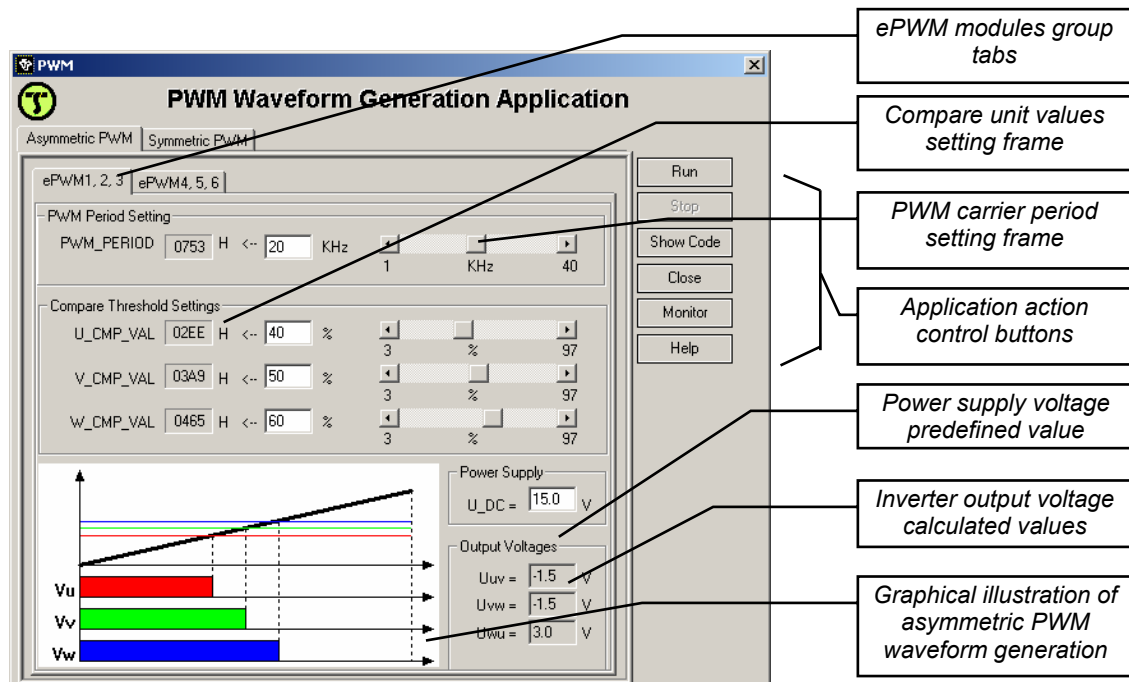


Figure 5.10. PWM Generation Application - Asymmetric PWM dialog

5.5.1.3 Actions to run the application

Step 1.

- Setup the PWM period in the PWM carrier period setting frame

Step 2.

- Setup the three compare values in the compare unit values setting frame

Step 3.

- Click on the **Run** button to start the experiment

Step 4.

- Use an oscilloscope for visualization of the 6 PWM outputs (pins 3÷8 on **J3** / **J4** connector)
- Repeat Step 2 with modified values while the application is running to see on-line effects on the generated voltages.

5.5.1.4 DSC software description

Figure 5.11 presents the flowchart of the program for asymmetric PWM generation.

5.5.1.5 Program files

The file name containing the DSC program for this application is **pwma.c**. Activating the Show code action control button in the application dialog can inspect it. The C header file called **pwma.h** holds the variables and function declarations for the application.

5.5.1.6 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**. Program variables are allocated in external data memory starting at address **0x3F9000**:

0x3F9000: DemoPWMA.stop	program stop index (set to 1 to end the program)
0x3F9001: DemoPWMA.pwm_period	variable containing the PWM carrier period to be loaded in Time Base Period Register (TBPRD)
0x3F9002: DemoPWMA.u_cmp_value	variable containing the value to be loaded in Compare A Register for ePWM1 / ePWM4
0x3F9003: DemoPWMA.v_cmp_value	variable containing the value to be loaded in Compare A Register for ePWM2 / ePWM5
0x3F9004: DemoPWMA.w_cmp_value	variable containing the value to be loaded in Compare A Register for ePWM3 / ePWM6
0x3F9005: DemoPWMA.demo_selection	variable specifying which ePWM modules group are to be used (ePWM1 , ePWM2 , ePWM3 or ePWM4 , ePWM5 , ePWM6)

5.5.1.7 Application results evaluation

The 6 PWM generated signals can be monitored with an oscilloscope. Alternatively, if an external inverter is connected to the MSK28335 kit on the MC-bus, the three-phased voltage realized by the asymmetric PWM technique can be generated at the output of the inverter bridge. The line voltages can be measured with a voltmeter in this case.

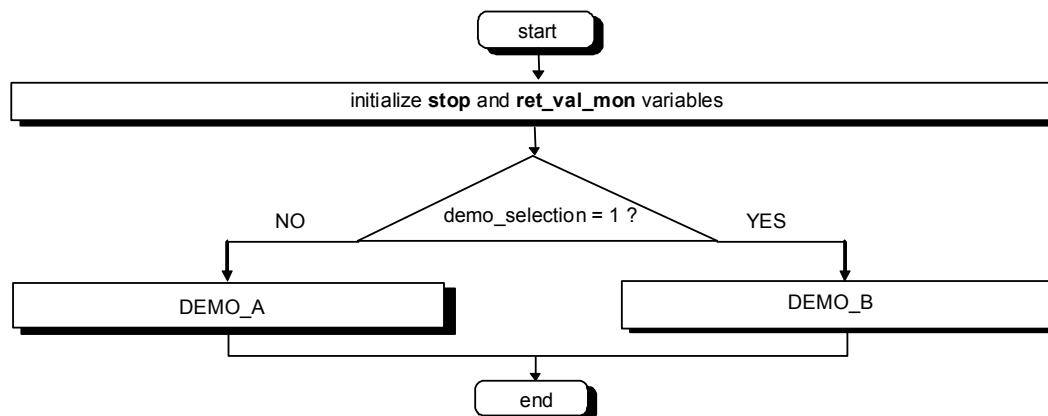


Figure 5.11.A. PWM Waveform Generation Application - Asymmetric PWM DSC program flowchart

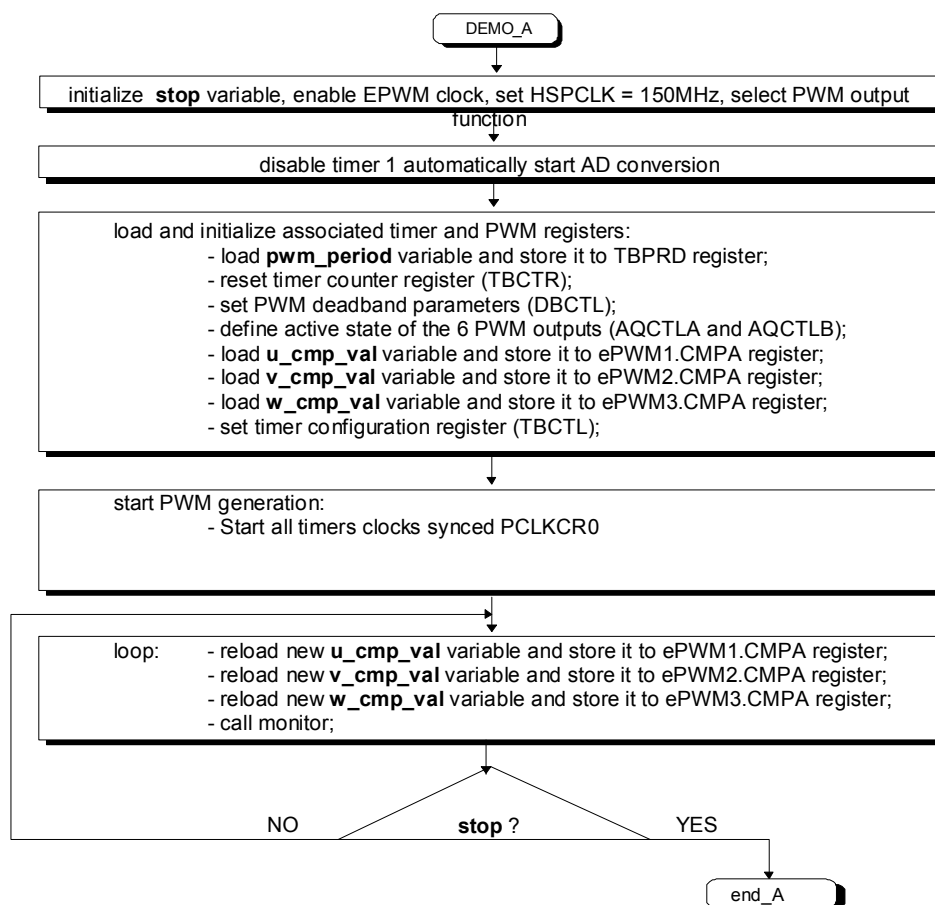


Figure 5.11.B. PWM Waveform Generation Application - Asymmetric PWM DSC program flowchart

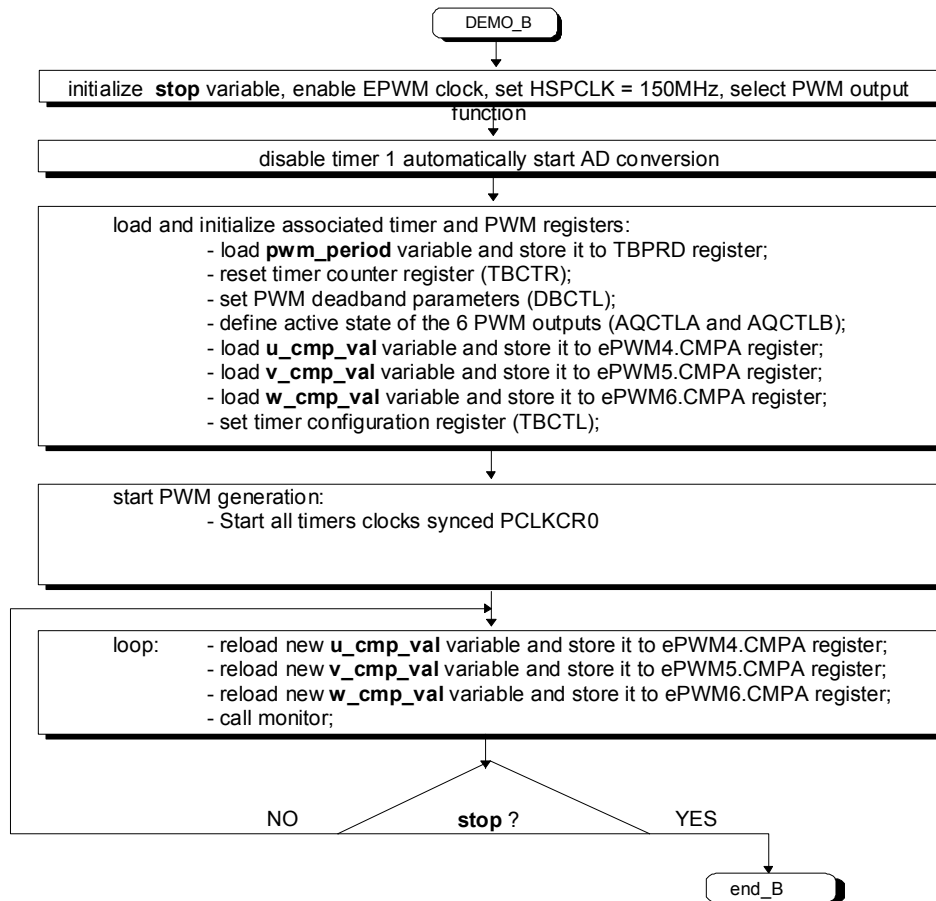


Figure 5.11.C. PWM Waveform Generation Application - Asymmetric PWM DSC program flowchart

5.5.2 Symmetric PWM

5.5.2.1 Purpose

Generate three-phase voltage using symmetric pulse width modulation (PWM) technique.

ePWM modules 1...3 or 4...6 of controller are programmed to give 6 PWM signals in PWM symmetric mode.

5.5.2.2 Screen Presentation

The screen presented in Figure 5.12 is accessible by selecting the **Symmetric PWM** panel in the **PWM Waveform Generation Application** dialog.

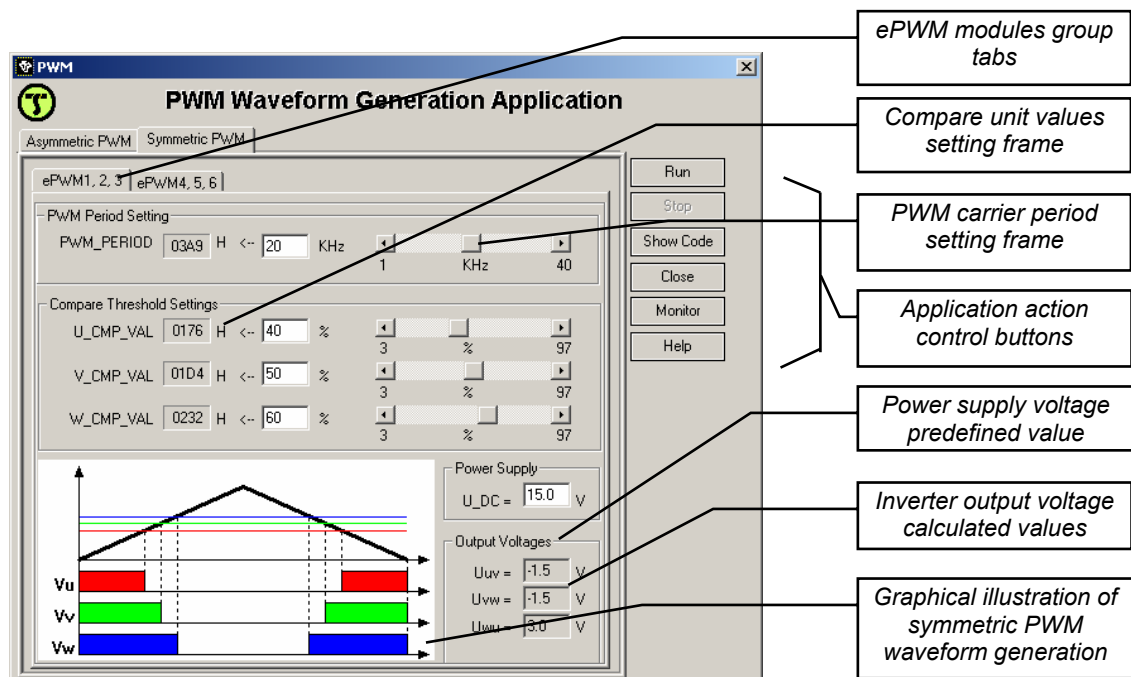


Figure 5.12. PWM Waveform Generation Application - Symmetric PWM dialog

5.5.2.3 Actions to run the application

Step 1.

- Set up the PWM period in the PWM carrier period setting frame

Step 2.

- Set up the three compare values in the compare unit values setting frame

Step 3.

- Click on the **Run** button to start the experiment

Step 4.

- Use an oscilloscope for visualization of the 6 PWM outputs (pins 3 - 8 on **J3** / **J4** connector)

Repeat Step 2 with modified values while the application is running to see on-line effects on the generated voltages.

5.5.2.4 Program files

The file name containing the DSC program for this application is **pwms.c**. Activating the Show code action control button in the application dialog can inspect it. The C header file called **pwms.h** holds the variables and function declarations for the application.

5.5.2.5 DSC software description

Figure 5.13 presents the flowchart of the program for symmetric PWM generation.

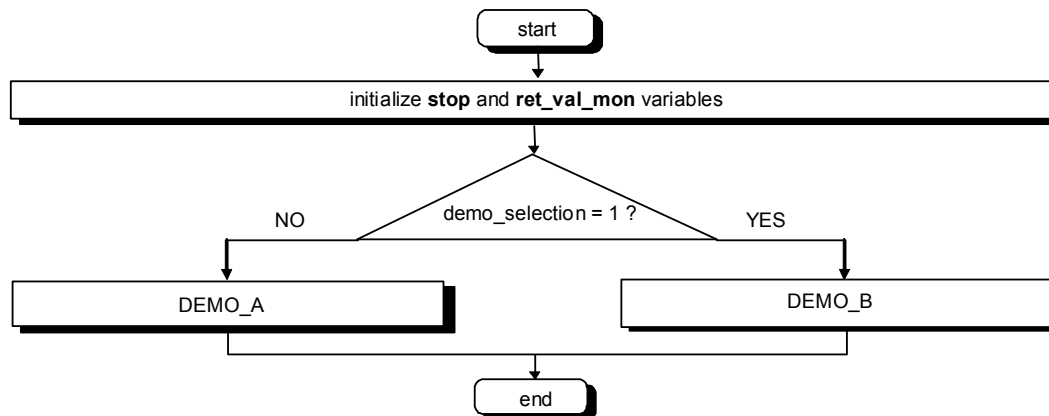


Figure 5.13.A. PWM Waveform Generation Application-Symmetric PWM DSC program flowchart

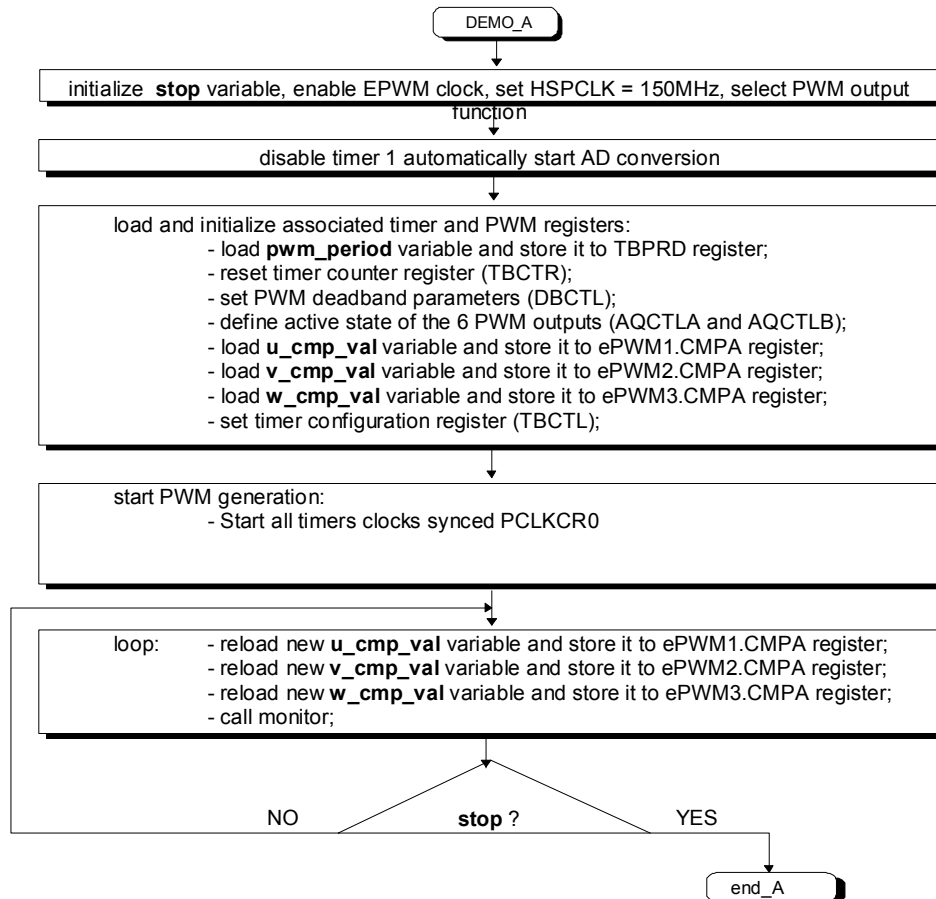


Figure 5.13.B. PWM Waveform Generation Application-Symmetric PWM DSC program flowchart

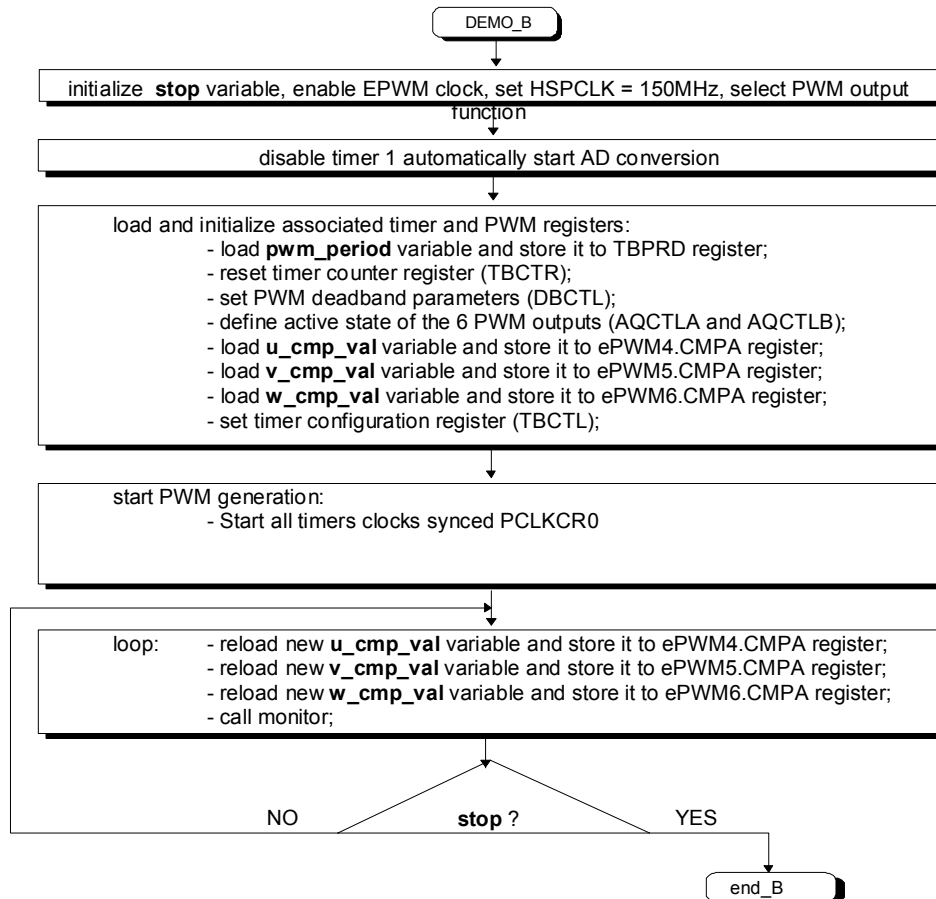


Figure 5.13.C. PWM Waveform Generation Application-Symmetric PWM DSC program flowchart

5.5.2.6 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**.

Program variables are allocated in external data memory starting at address **0x3f9000**:

0x3F9000: DemoPWMS.stop	program stop index (set to 1 to end the program)
0x3F9001: DemoPWMS.pwm_period	variable containing the PWM carrier period to be loaded in Time Base Period Register (TBPRD)
0x3F9002: DemoPWMS.u_cmp_value	variable containing the value to be loaded in Compare A Register for ePWM1 / ePWM4
0x3F9003: DemoPWMS.v_cmp_value	variable containing the value to be loaded in Compare A Register for ePWM2 / ePWM5
0x3F9004: DemoPWMS.w_cmp_value	variable containing the value to be loaded in Compare A Register for ePWM3 / ePWM6
0x3F9005: DemoPWMS.demo_selection	variable specifying which ePWM modules group are to be used (ePWM1 , ePWM2 , ePWM3 or ePWM4 , ePWM5 , ePWM6)

5.5.2.7 Application results evaluation

The 6 PWM generated signals can be monitored with an oscilloscope. Alternatively, if an external inverter is connected to the MSK28335 kit on the MC-bus, the three-phased voltage realized by the symmetric PWM technique can be generated at the output of the inverter bridge. The line voltages can be measured with a voltmeter.

5.6 Quadrature Encoder Pulse (QEP) Circuit Application

5.6.1 Purpose

Measure and store position information from an incremental encoder using the eQEP Circuits. This demo can be used properly only if you connect a quadrature encoder on the eQEP interfaces inputs.

The Quadrature Encoder Pulse Circuit is enabled to decode and count the quadrature encoded input pulses on **GPIO50 / EQEP1A / XD29** and **GPIO51 / EQEP1B / XD28** pins for **eQEP1** and on **GPIO24 / ECAP1 / EQEP2A / MDXB** and **GPIO25 / ECAP2 / EQEP2B / MDRB** pins for **eQEP2**. Encoder shaft movement is graphically illustrated in the application dialog.

5.6.2 Screen Presentation

Selecting the « **QEP** » button in the **Processor Evaluation Control Panel** menu activates the **Quadrature Encoder Pulse (QEP) Circuit Application** dialog presented in Figure 5.16.

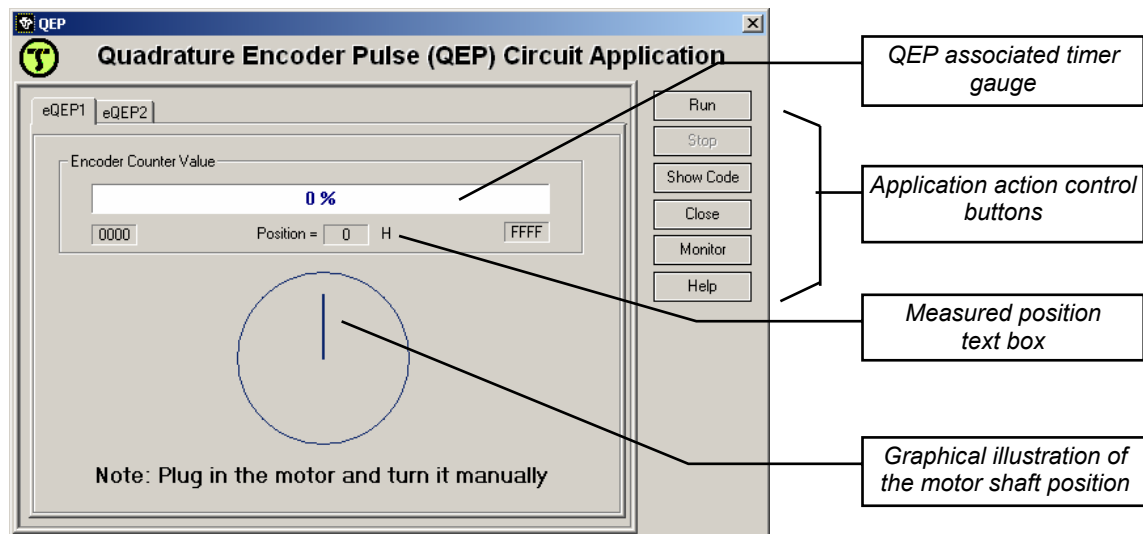


Figure 5.16. Quadrature Encoder Pulse (QEP) Circuit Application dialog

5.6.3 Actions to run the application

Step 1.

- Connect quadrature encoder signals A and B to corresponding inputs of the power module.

Step 2.

- Click on the **Run** button to start the experiment.

Step 3.

- Rotate the encoder's shaft to see the measured position value updated on the screen.

5.6.4 DSC software description

Programming the DSC to perform this experiment is illustrated in the program flowchart (fig 5.17)

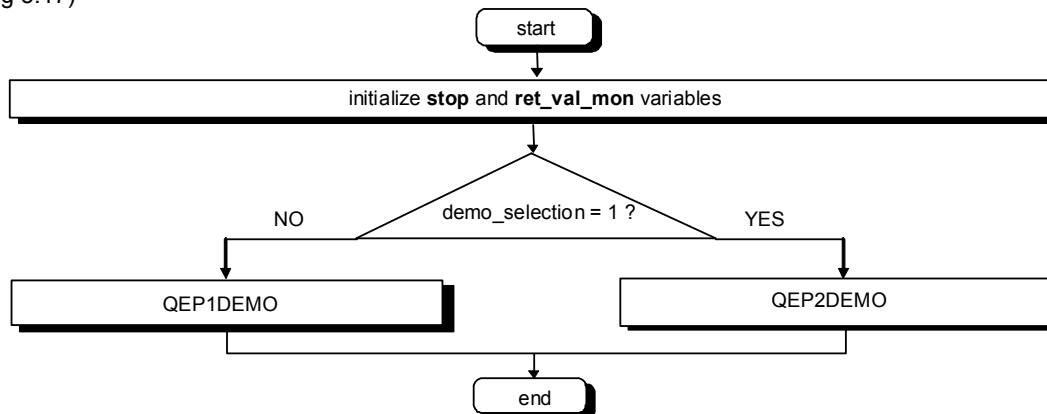


Figure 5.17.A. Quadrature Encoder Pulse (QEP) Circuit Application DSC program flowchart

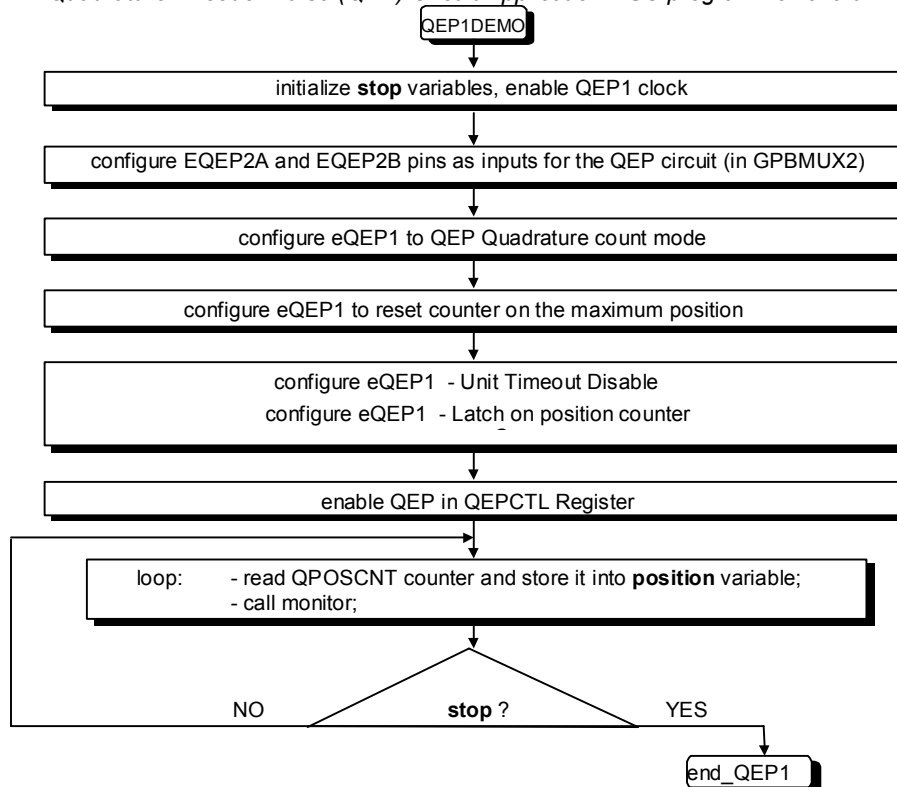


Figure 5.17.B. Quadrature Encoder Pulse (QEP) Circuit Application DSC program flowchart

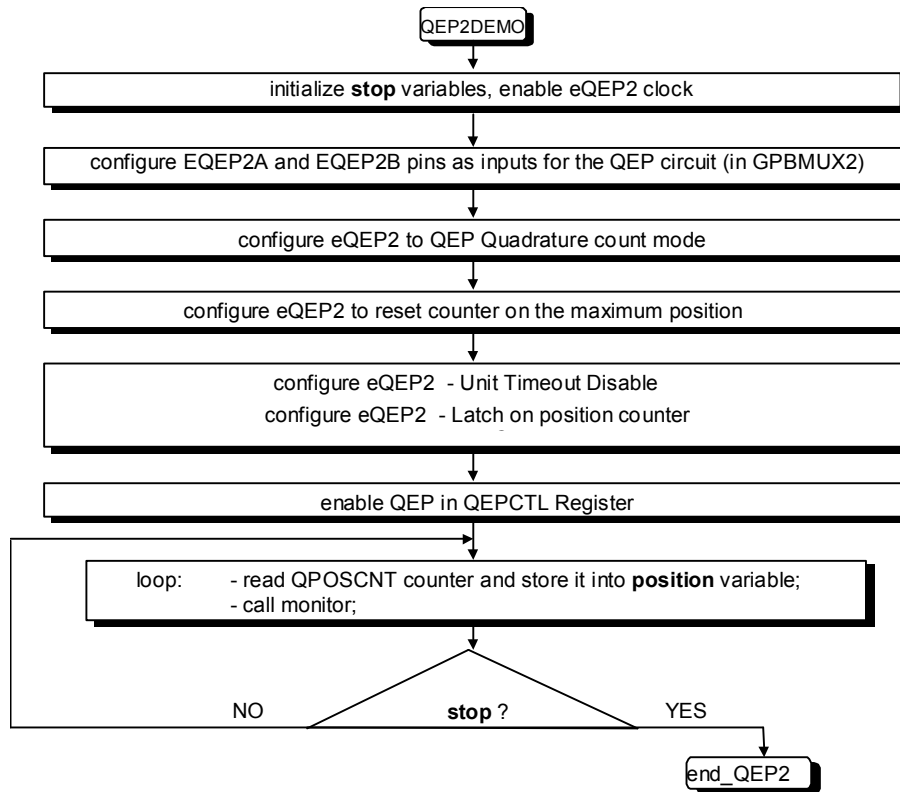


Figure 5.17.C. Quadrature Encoder Pulse (QEP) Circuit Application DSC program flowchart

5.6.5 Program files

The file **qep.c** contains the DSC program described in Figure 5.17. Activating the Show code action control button in the application dialog can inspect it. The C header file called **qep.h** holds the variables and function declarations for the application.

5.6.6 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**. Program variables are allocated in external data memory starting at address **0x3F9000**:

0x3F9000: DemoQEP.stop	program stop index (set to 1 to end the program)
0x3F9001: DemoQEP.position	encoder measured position obtained as the last value read from the eQEP1 / eQEP2 position counter
0x3F9002: DemoQEP.module_selection	variable specifying which eQEP module are to be used

5.6.7 Application results evaluation

The application controls the QEP circuit to obtain position information from an incremental encoder. The result can be evaluated in the **Quadrature Encoder Pulse (QEP) Circuit Application** dialog as:

- graphical illustration of the encoder shaft position
- hexadecimal value in the measured position text box
- red gauge representing the evolution of the eQEP1 / eQEP2 position counter

5.7 Serial Communication Interface (SCI) Module Application

5.7.1 Purpose

Program and control a serial communication between the PC and the **MSK28335**.

Strings of characters can be sent / received with programmed baud rate to/from the **MSK28335** board. User can select a standard baud rate to set-up a serial communication between the PC and the **MSK28335**. Characters entered on the PC keyboard are displayed on the PC Transmit buffer and send to the **MSK28335**. The board echoes them and the returned characters can be inspected in the PC Receive buffer.

5.7.2 Screen Presentation

The dialog presented in Figure 5.18 is accessible by selecting the « **SCI** » button in the **Processor Evaluation Control Panel** menu.

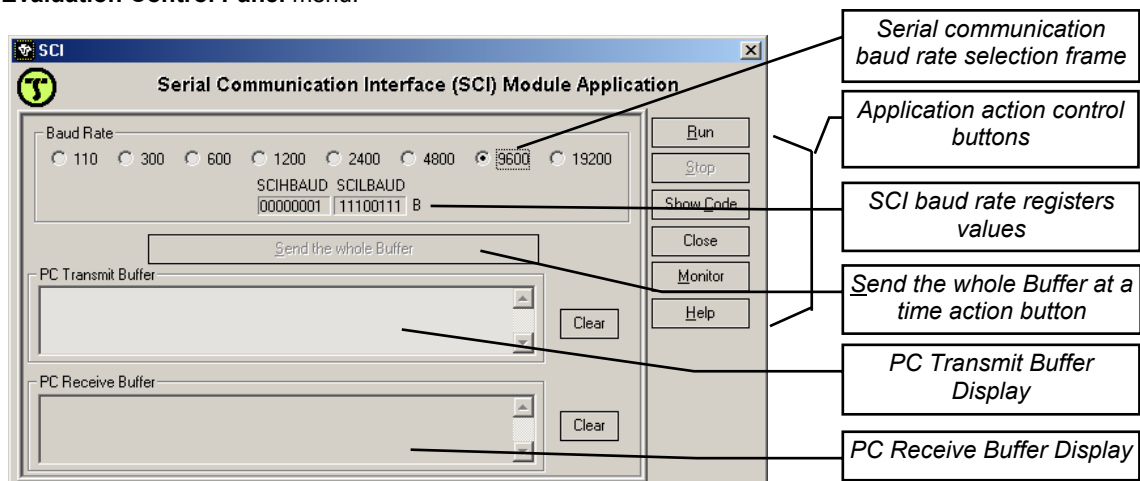


Figure 5.18. Serial Communication Interface (SCI) Module Application dialog

5.7.3 Actions to run the application

Step 1.

- Click on a button in the “**Serial communication baud rate selection**” frame to set-up the desired serial baud rate.

Step 2.

- Click on the **Run** button to start the experiment.

Step 3.

- Enter characters from the keyboard. As they are entered, the characters are displayed on the PC Transmit Buffer Display, sent to the **MSK28335** board, received and echoed back to the PC by the DSC application program and displayed on the PC Receive Buffer Display. Both Transmit and Receive buffers can be cleared at any time with the associated **Clear** action button while the application is running. The whole string stored in the PC Transmit Buffer can be sent at a time with the **Send the whole Buffer** action button.

5.7.4 DSC software description

Figure 5.19 presents the flowchart of the DSC program that performs this application.

5.7.5 Program files

The file name containing the DSC program for this application is **sci.c**. Activating the Show code action control button in the application dialog can inspect it. The C header file called **sci.h** holds the variables and function declarations for the application.

5.7.6 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**, and two routine functions: **ci** and **co**.

Program variables are allocated in external data memory starting at address **0x3F9000**:

0x3F9000: DemoSCI.scihbaud	variable containing the value for SCIHBAUD Register
0x3F9001: DemoSCI.scilbaud	variable containing the value for SCILBAUD Register
0x3F9002: DemoSCI.character	variable containing the received/transmitted character

5.7.7 Application results evaluation

The application sets-up a pooling-type serial communication link between the PC and the DSC board. Characters entered by the keyboard are sent to the DSC board and echoed back to the PC. Transmit and receive buffers special displays are available in the application dialog. Inspecting DSC program code helps understanding the programming of the on-chip Serial Communication Interface Module.

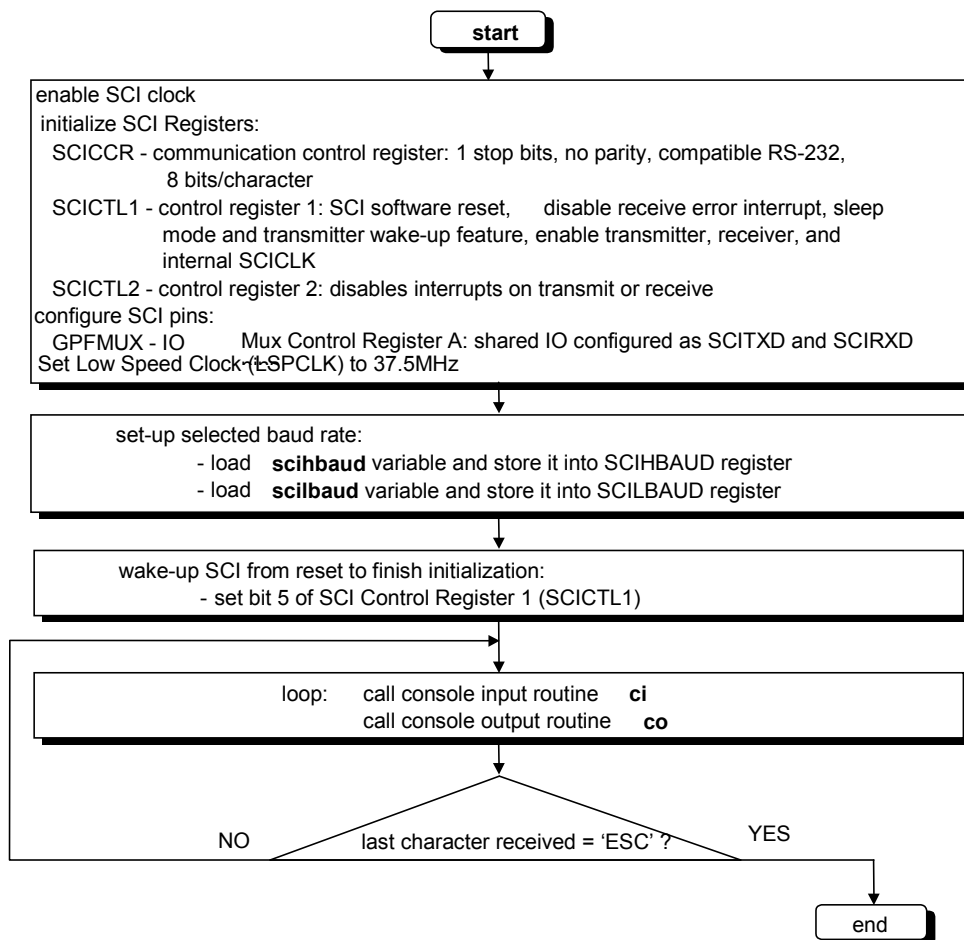


Figure 5.19.A. Serial Communication Interface (SCI) Module Application DSC program flowchart

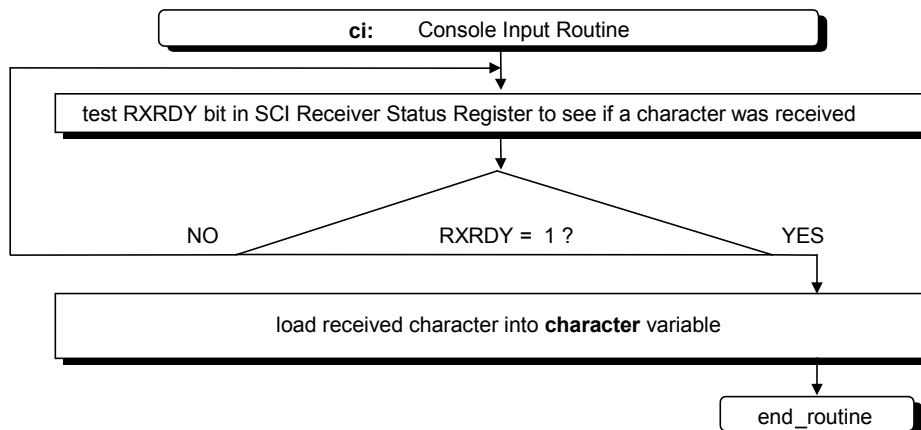


Figure 5.19.B. Serial Communication Interface (SCI) Module Application DSC program flowchart

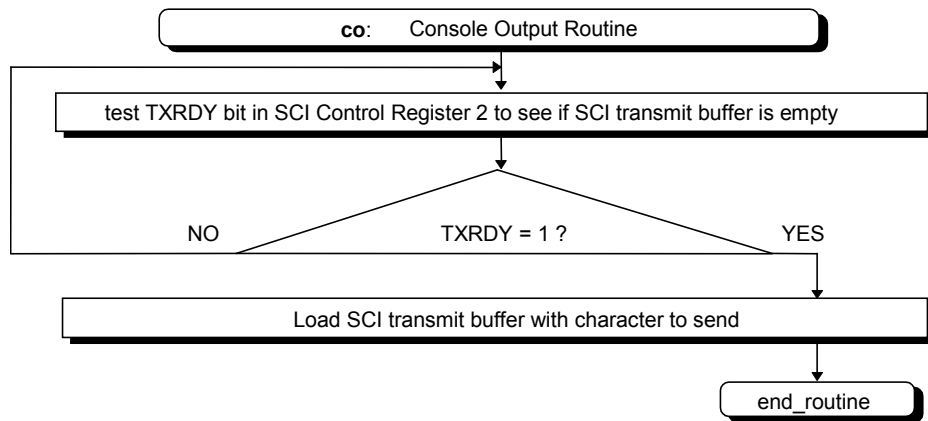


Figure 5.19.C. Serial Communication Interface (SCI) Module Application DSC program flowchart

5.8 Serial Peripheral Interface (SPI) Module Application

5.8.1 Purpose

Send/receive user specified characters codes to/from the high-speed on-chip Serial Peripheral Interface. The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communication between the DSC controller and external peripherals or another controller.

This application programs the SPI to send/receive a 16-bit stream representing a code entered by you in the application dialog. External connection is needed to route the SPI output pin back to the SPI input pin or, as an alternative, to connect the interface to a similar SPI interface of another controller.

5.8.2 Screen Presentation

The dialog presented in Figure 5.20 is accessible by selecting the « **SPI** » button in the **Processor Evaluation Control Panel** menu.

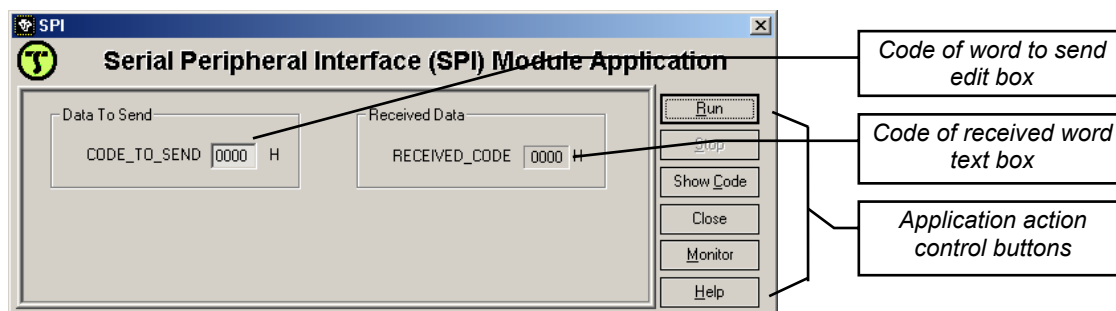


Figure 5.20. Serial Peripheral Interface (SPI) Module Application dialog

5.8.3 Actions to run the application

Step 1.

- Enter a code composed of four hexadecimal values in the CODE_TO_SEND edit box.

Step 2.

- Click on the **Run** button to start the experiment.

Step 3.

- The SPISIMO pin of the SPI is internally routed to the SPISOMI pin, using loopback mode of SPI module. The 16-bit length stream, representing the code to send, will be echoed by SPI and will appear in the REC_CODE text box.

5.8.4 DSC software description

Figure 5.21 presents the flowchart of the DSC program that performs this application.

5.8.4.1 Program files

The file name containing the DSC program for this application is **spi.c**. Activating the Show code action control button in the application dialog can inspect it. The assembler header file called **spi.h** holds the variables and function declarations for the application.

5.8.5 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**, and two routine functions: **send** and **receive**.

Program variables are allocated in external data memory starting at address **0x3F9000**:

0x3F9000: DemoSPI.stop	program stop index (set to 1 to end the program)
0x3F9001: DemoSPI.code_to_send	16 bit value representing the code of a word to be sent on the SPI serial line
0x3F9002: DemoSPI.rec_code	16 bit value representing the code of a word received on the SPI serial line

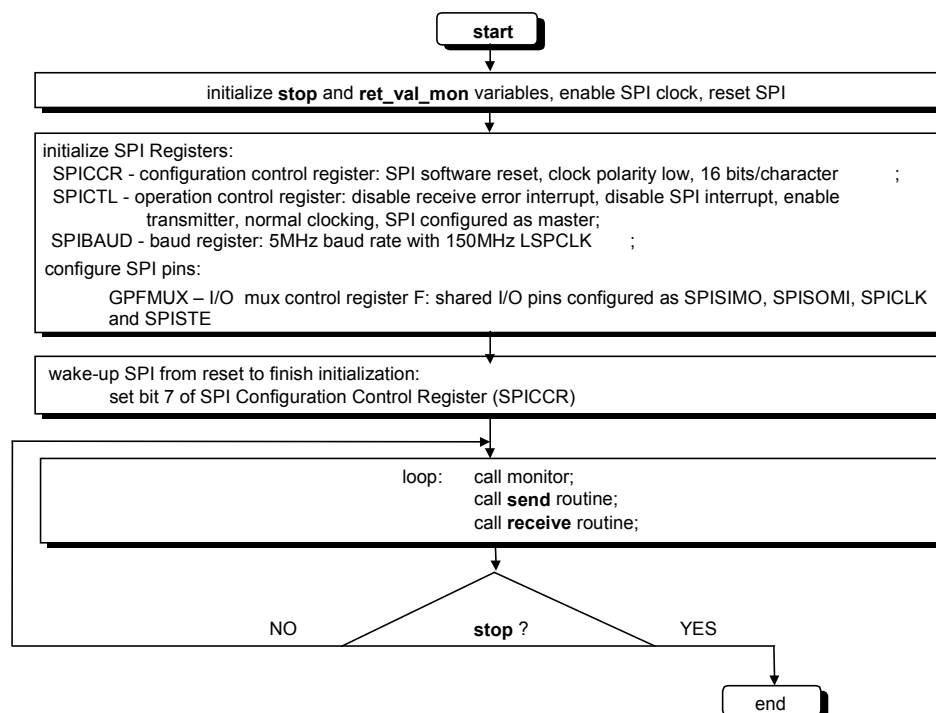


Figure 5.21.A. Serial Peripheral Interface (SPI) Module Application DSC program flowchart

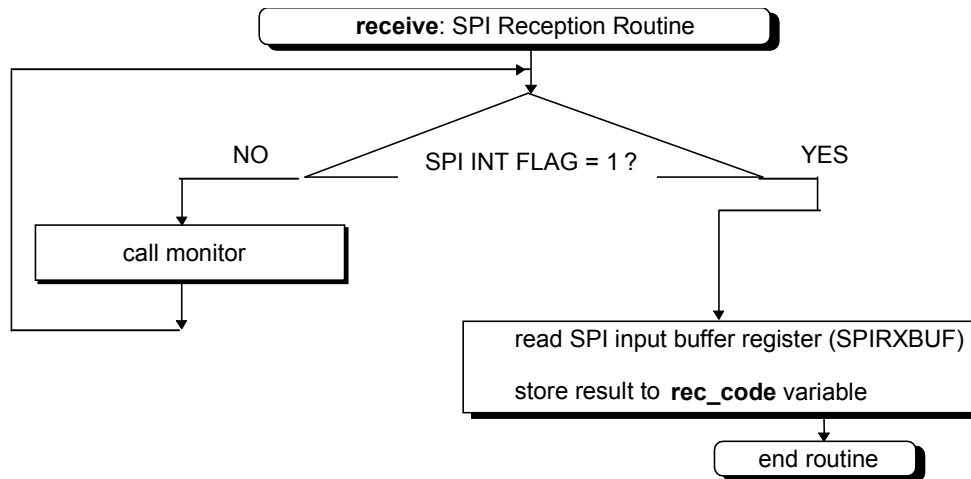


Figure 5.21.B. Serial Peripheral Interface (SPI) Module Application DSC program flowchart

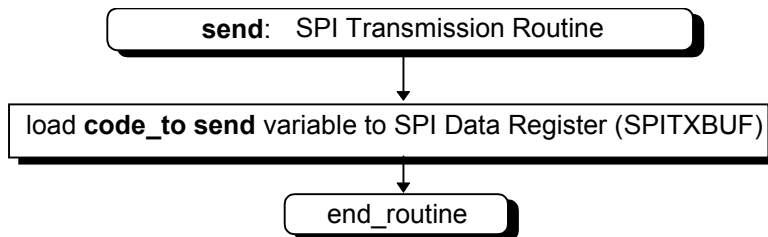


Figure 5.21.C. Serial Peripheral Interface (SPI) Module Application DSC program flowchart

5.8.6 Application results evaluation

The Serial Peripheral Interface is programmed in this application to continuously transmit/receive 16 bit streams on the input/output serial lines. The transmitted values are entered by you in the application dialog. An external connection can easily be made to route back the serial output to serial input SPI lines. Received codes are then identical with the transmitted ones. The application dialog displays also the code received on the SPI serial input line. Inspecting DSC program code helps understanding the programming of the on-chip Serial Peripheral Interface Module.

5.9 Enhanced Controller Area Network (eCAN) Module Application

5.9.1 Purpose

This module configures the on-chip CAN controller for sending and receiving one CAN message via CAN peripheral interface.

The on-chip CAN peripheral interface may operate in two modes:

- In self test mode when the message is not sent but read back and stored in the appropriate mailbox.
- In network mode when the message is sent via the CAN bus to other CAN node. In this mode the two (or more) CAN nodes must be configured with the same timing parameters.

Two CAN mailboxes are used: mailbox 0 for reception and mailbox 5 for transmission of the CAN messages. It is recommended to configure the reception mailbox before the transmission mailbox, so that the sent message can be received back correctly. During the communication process, the reception and transmission statuses are displayed on-line. Also, the Error and Status Register is on-line monitored, to allow an easy identification of errors' source.

5.9.2 Screen Presentation

The dialog presented in Figure 5.22 is accessible by selecting the « **CAN** » button in the **Processor Evaluation Control Panel** menu.

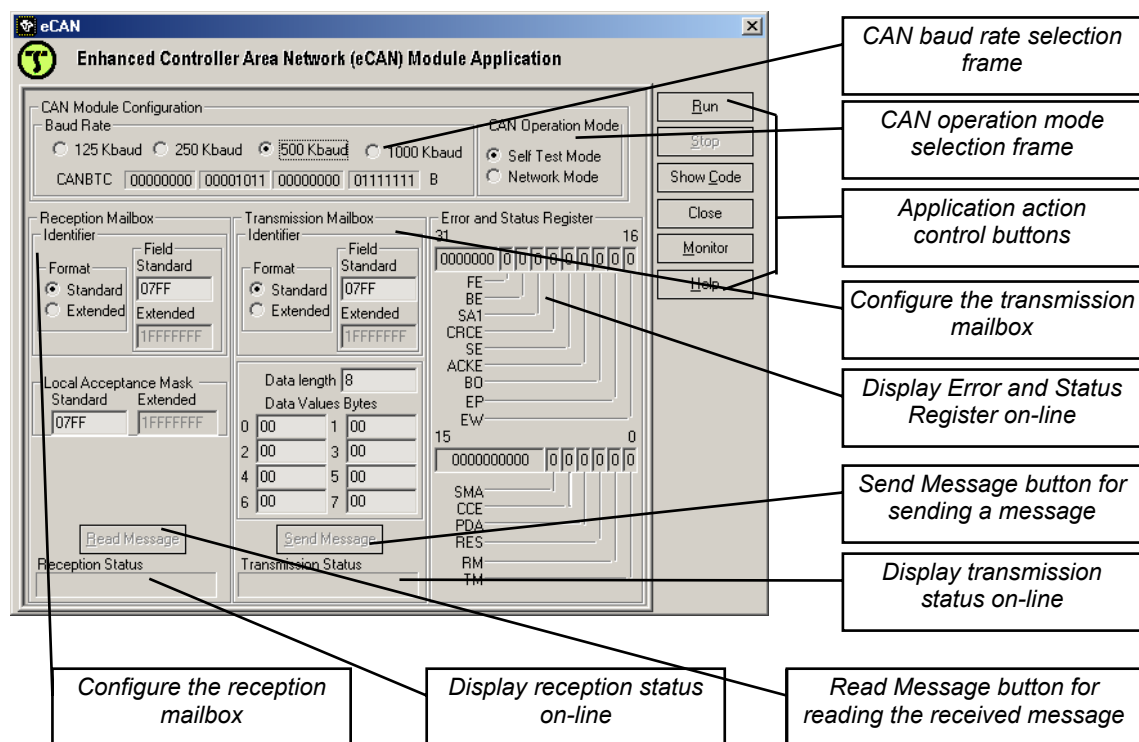


Figure 5.22. CAN Controller Module Application dialog

5.9.3 Actions to run the application

Step 1.

- Click on a button in the CAN baud rate selection frame to set-up the desired serial baud rate.

Step 2.

- Click on a button CAN operation mode selection frame to specify the operation mode.

Step 3.

- Configure the reception mailbox by selecting:
 - the identifier format: standard (11 bits) or extended (29 bits);
 - the identifier field: contains the hexadecimal value of the message identifier;
 - the local acceptance mask(hexadecimal value):
 - “1” means that don’t care the value for that bit position,
 - “0” means that the incoming bit value must be identical to the corresponding bit in the message identifier field.

Step 4.

- Configure the transmission mailbox by selecting:
 - the identifier format: standard (11 bits) or extended (29 bits);
 - the identifier field: contains the hexadecimal value of the message identifier;
 - the data length: specifies the number of bytes which have to be transmitted;
 - the data values: contains the data which have to be transmitted via CAN bus.

Step 4.

- Click on the **Run** button to start the experiment.

Step 5.

- Click on the Send Message button to send the information contained in the transmission mailbox.

Step 6.

- Click on the Read Message button to read the message arrived in the reception mailbox, if any. The Read Message button is active only if the message successfully arrives to destination. In case of error, the Error and Status Register displays the source of error, and the user has to stop the demonstration.

5.9.4 DSC software description

Figure 5.23 presents the flowchart of the DSC program that performs this application.

5.9.5 Program files

The file name containing the DSC program for this application is **can.c**. Activating the Show code action control button in the application dialog can inspect it. The C header file called **can.h** holds the variables and function declarations for the application.

5.9.6 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**, and two routine functions: **SendMessage**, and **ReadMessage**.

Program variables are allocated in external data memory starting at address **0x3F9000**:

0x3F9000: DemoCAN.stop	program stop index (set to 1 to end the program)
0x3F9001: DemoCAN.op_mode	variable indicating the operation mode: 1=Self Test Mode, 0=Network Mode
0x3F9002: DemoCAN.bit_tim_param	long variable containing the value for BCT Register
0x3F9004: DemoCAN.rx_format_id	variable indicating the identifier format for reception mailbox: standard (0) or extended (1)
0x3F9005: DemoCAN.rx_std_id	variable containing the standard identifier value for reception mailbox
0x3F9006: DemoCAN.rx_xtd_id	long variable containing the extended identifier value for reception mailbox
0x3F9008: DemoCAN.tx_format_id	variable indicating the identifier format for transmission mailbox: standard (0) or extended (1)
0x3F9009: DemoCAN.tx_std_id	variable containing the standard identifier value for transmission mailbox
0x3F900a: DemoCAN.tx_xtd_id	long variable containing the extended identifier value for transmission mailbox
0x3F900c: DemoCAN.std_accept_mask	variable containing the value of the standard acceptance mask
0x3F900e: DemoCAN.xtd_accept_mask	long variable containing the value of the extended acceptance mask
0x3F9010: DemoCAN.tx_data_len	variable containing the length of the data which are to be transmitted
0x3F9011: DemoCAN.tx_data_val	variable containing the buffer of data which are to be transmitted
0x3F9015: DemoCAN.rx_data_len	variable containing the length of the data which are to be received
0x3F9016: DemoCAN.rx_data_val	variable containing the buffer of the data which are to be received
0x3F901a: DemoCAN.data_len	variable containing the length of data in words
0x3F901b: DemoCAN.tx_msg_rq	variable indicating if the Send Message button was activated
0x3F901c: DemoCAN.rx_msg_rq	variable containing if the Read Message button was activated

5.9.7 Application results evaluation

The application allows testing of the CAN communication in two operation modes. In self-test mode the board receives back the message sent by the board itself. In network mode, the board can communicate with other board, if the timing parameters of both boards are identically. For both operation modes, the transmitter identifier must be identical with the reception identifier, for all null bits of the local acceptance mask.

After setting the CAN global registers (baud rate and operation mode) and configuration of reception and transmission mailboxes, a message may be sent. If the message arrives to the destination (in the reception mailbox), it may be read and compared with the transmitted message. The messages must be identical.

If an error occurs during communication, the error's source is displayed in the Error and Status Register frame. The statuses of the transmission or reception operations are also monitored. Inspecting DSC program code helps understanding the programming of the on-chip CAN Interface Module.

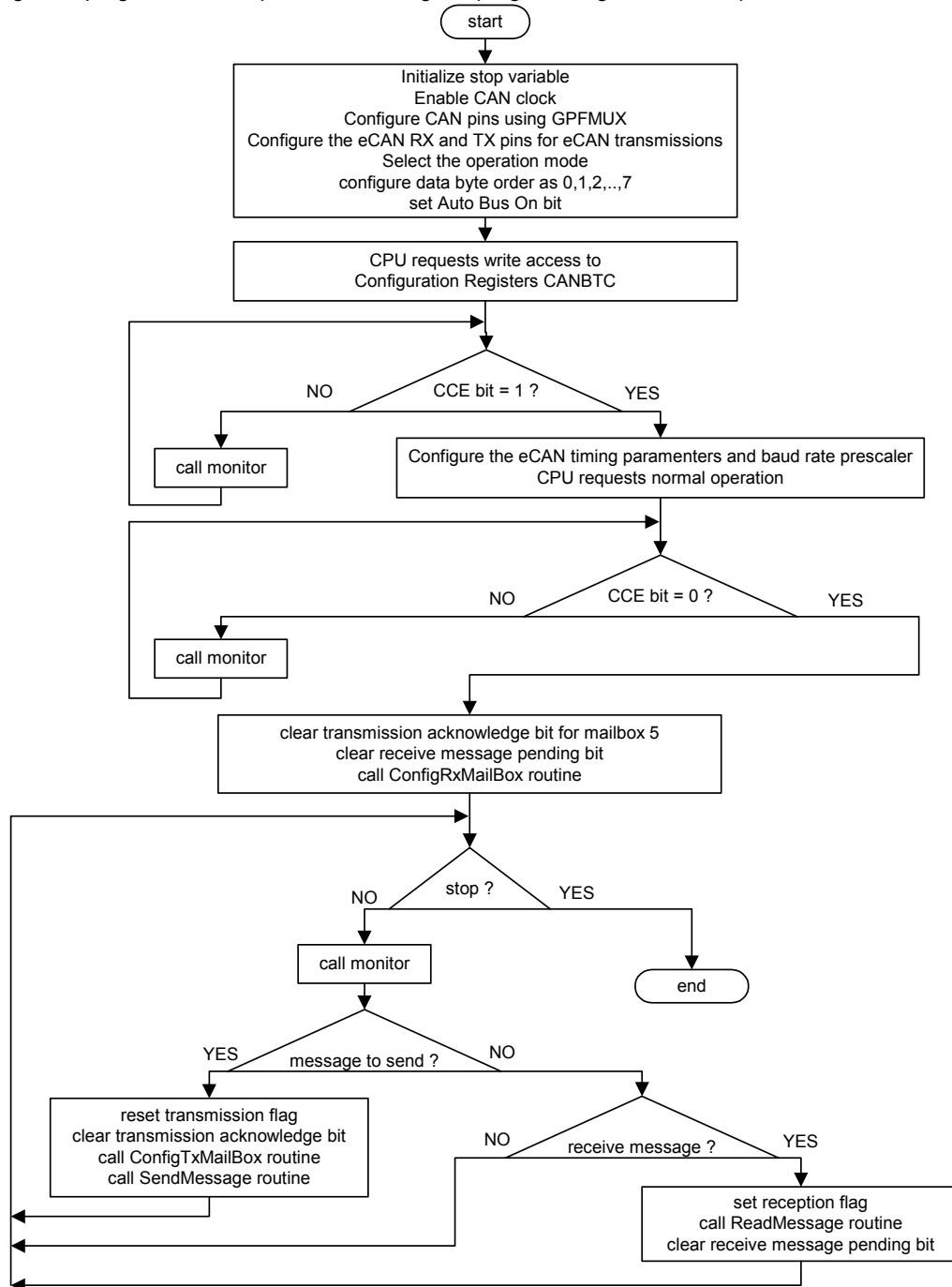
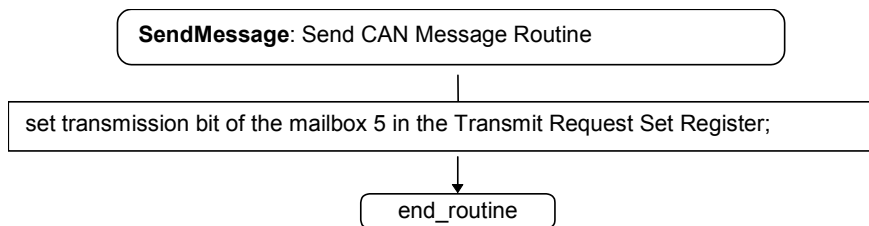
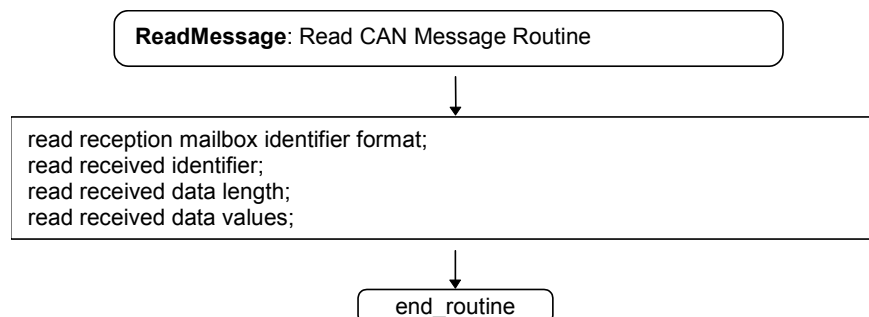
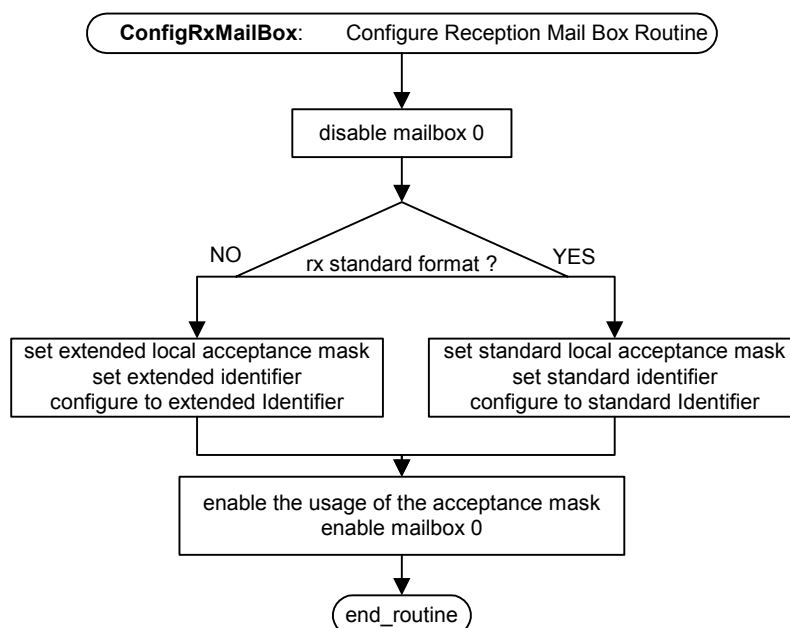


Figure 5.23.A. CAN Controller Module Application DSC program flowchart**Figure 5.23.B.** CAN Controller Module Application DSC program flowchart**Figure 5.23.C.** CAN Controller Module Application DSC program flowchart**Figure 5.23.D.** CAN Controller Module Application DSC program flowchart

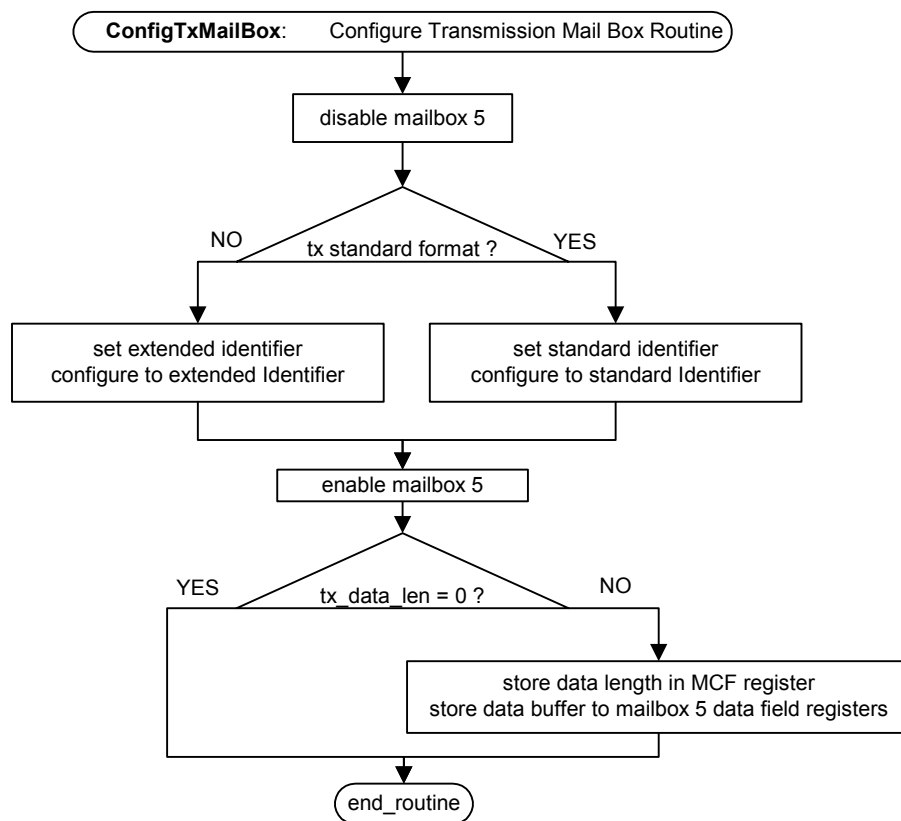


Figure 5.23.E. CAN Controller Module Application DSC program flowchart

5.10 General Purpose Timer (GPT) Application

5.10.1 Basic Timer Programming

5.10.1.1 Purpose

Experience some of the features (counting mode, compare/PWM output generation) offered by the Timers of ePWM modules.

The timer can be configured in any of the possible counting modes with internal clock using 2 sets of 8 prescale ratios. The compare output of the timer is enabled to be active low when a compare match occurs.

5.10.1.2 Screen Presentation

The dialog presented in Figure 5.24 is accessible by selecting the « **Timers** » button in the **Processor Evaluation Control Panel** menu.

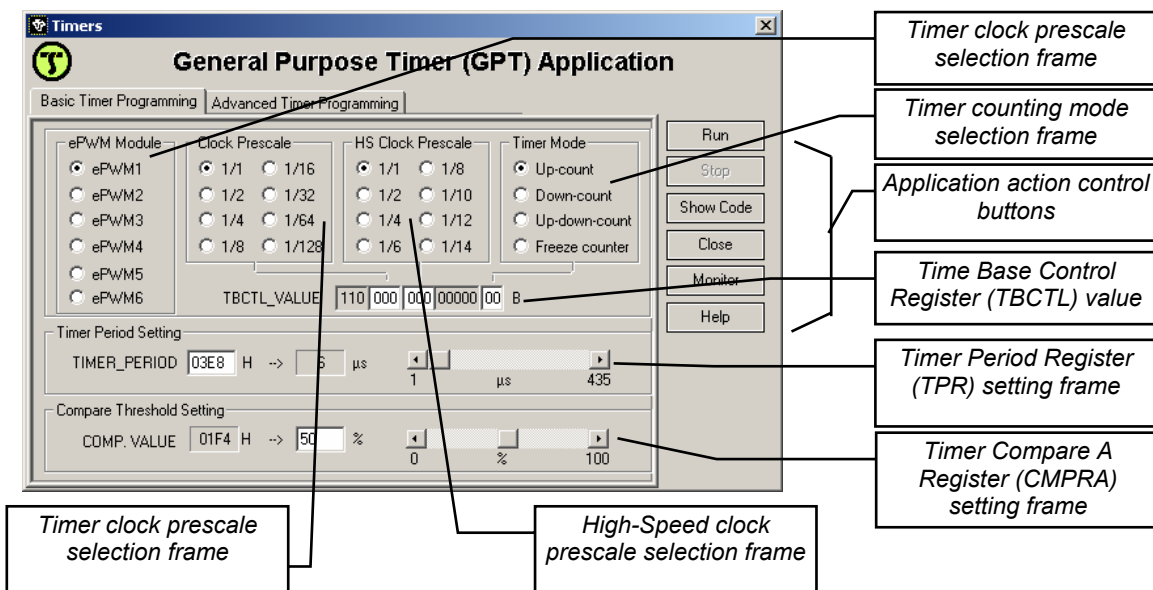


Figure 5.24. General Purpose Timer (GPT) Application - Basic Timer Programming dialog

5.10.1.3 Actions to run the application

Step 1.

- Select the timer mode in the Timer Mode selection frame

Step 2.

- Select the timer clock prescale in the Clock Prescale and HS Clock Prescale selection frames

Step 3.

- Fix the counting period of the timer, in the Timer Period Register (TBPRD) setting frame. Use one of the following features:
 - Write the hexadecimal value in the TIMER_PERIOD edit box, or

- Drag the cursor of the scroll bar to set the period in microseconds

Step 4.

- Set the compare value, in the Compare A Register (CMPA) setting frame. Use one of the following features:
 - Write the value (in percents of timer period) in the COMP.VALUE edit box, or
 - Drag the cursor of the scroll bar to set the compare value in percents of timer period

Step 5.

- Click on the **Run** button to start the experiment

Step 6.

- Use an oscilloscope for visualization of the compare output pin - EPWMxA / EPWMxB pins (pins 3...8 on **J3** connector or pins 3...8 on **J4** connector) according to the selected ePWM module.

5.10.1.4 DSC software description

Figure 5.25 presents the flowchart of the DSC program that performs this application.

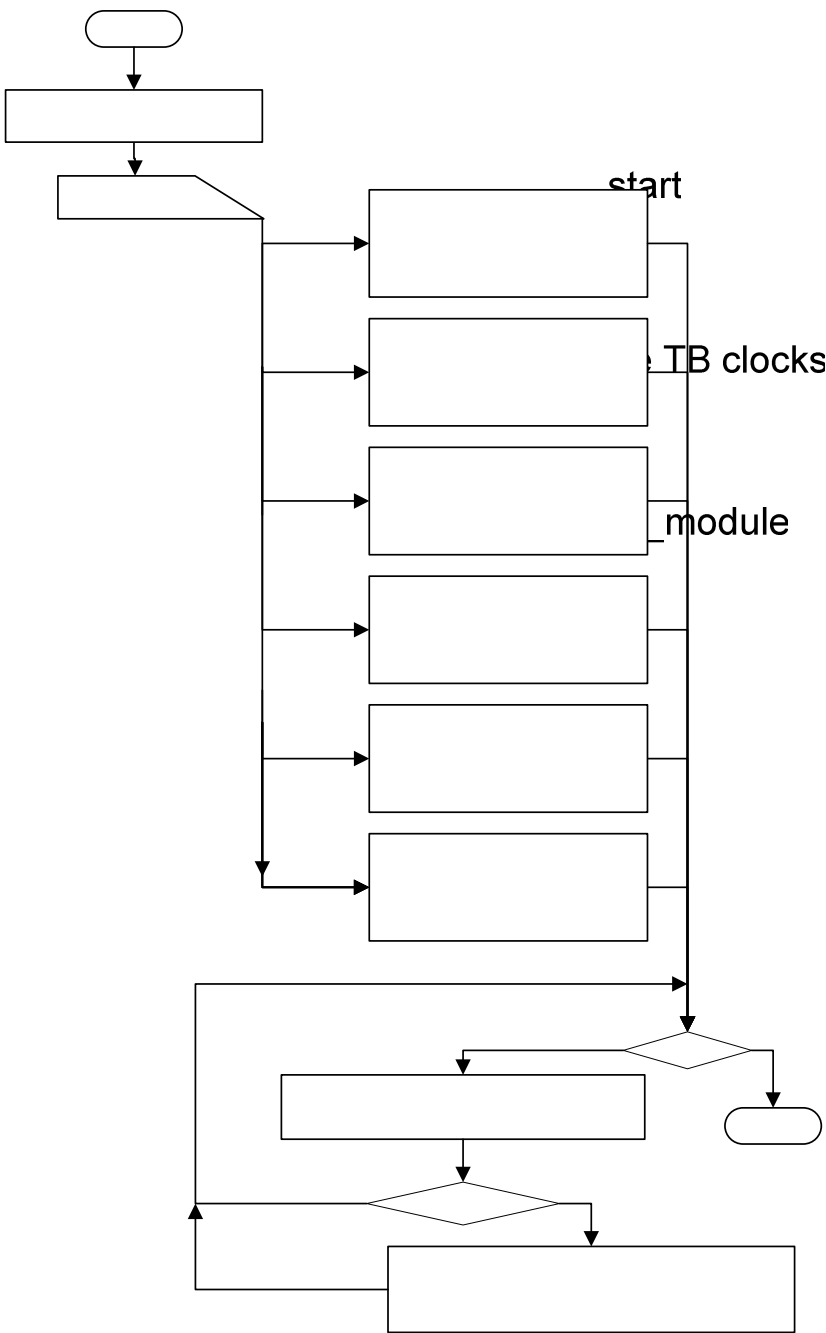


Figure 5.25.A. General Purpose Timer (GPT) Application - Basic Timer Programming DSC program flowchart

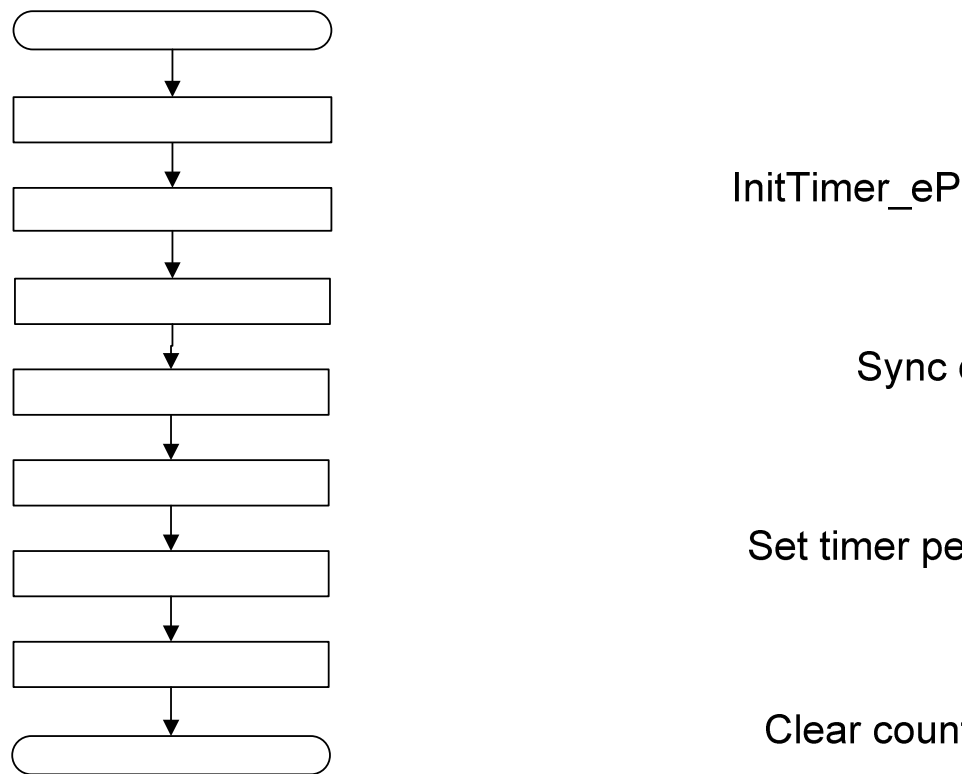


Figure 5.25.B. General Purpose Timer (GPT) Application - Basic Timer Programming DSC program flowchart

5.10.1.5 Program files

The file name containing the DSC program for this application is **gpt_1.c**. Activating the Show code action control button in the application dialog can inspect it. The C header file called **gpt_1.h** holds the variables and function declarations for the application.

5.10.1.6 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**. Program variables are allocated in external data memory starting at address **0x3F9000**:

0x3F9000: DemoGPT1.stop	program stop index (set to 1 to end the program)
0x3F9001: DemoGPT1.tbctl_value	variable containing the value for Time-Base Control Register (TBCTL)
0x3F9002: DemoGPT1.timer_period	variable containing the value for timer period register (TBPRD)
0x3F9003: DemoGPT1.compare_value	variable containing the value for Compare A register (CMPA)
0x3F9004: DemoGPT1.selected_module	variable specifying which ePWM module are to be used

5.10.1.7 Application results evaluation

A PWM signal based on the compare output of a general-purpose timer can be generated with this application. The frequency of the signal can vary between 1 microseconds (both prescales **1/1** and timer period value **c8h**) and 59 milliseconds (Clock prescale = **1/128** HS Clock prescale = **1/14** and timer period value **fff0h**). The signal's duty cycle can be configured from 0 to 100%.

5.10.2 Advanced Timer Programming

5.10.2.1 Purpose

Experience some the possibilities offered by the ePWM's Timers to generate interrupts to the processor.

ePWM1...6 modules can be selected for this application. It can be configured in any of the possible counting modes with internal DSC clock using 2 sets of 8 pre-scale ratios. The compare output of the timer is enabled to be active low when a compare match occurs.

5.10.2.2 Screen Presentation

The screen presented in Figure 5.26 is accessible by selecting the **Advanced Timer Programming** panel in the **General Purpose Timer (GPT) Application** dialog.

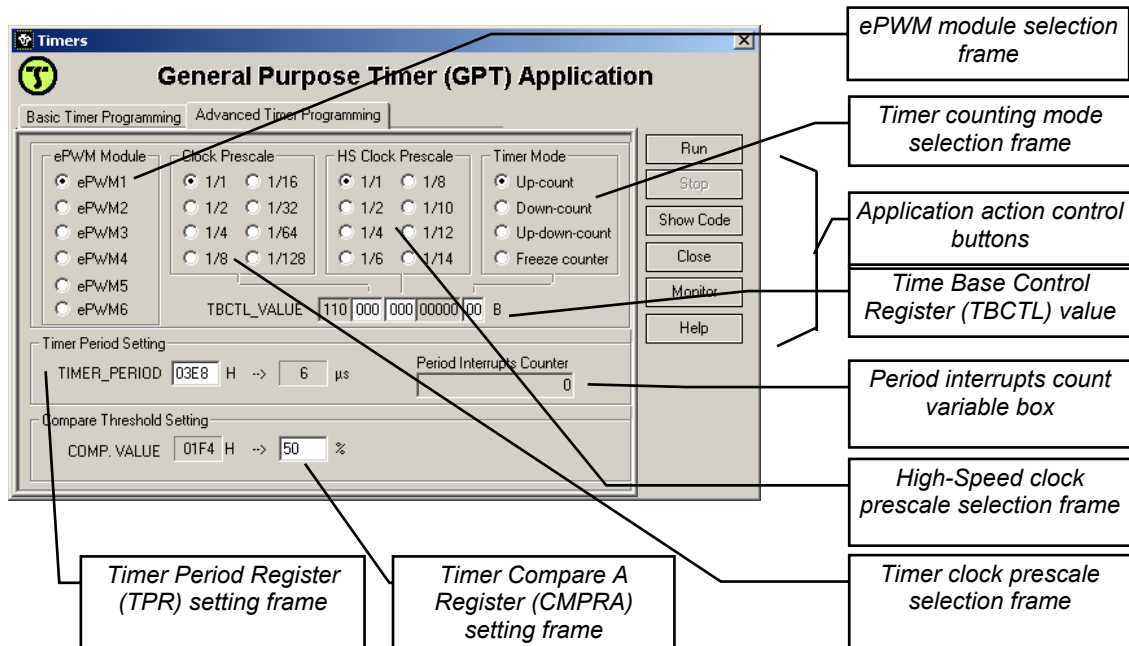


Figure 5.26. General Purpose Timer (GPT) - Advanced Timer Programming dialog

5.10.2.3 Actions to run the application

Step 1.

- Select the timer mode in the Timer Mode selection frame

Step 2.

- Select the timer clock prescale in the Clock Prescale and HS Clock Prescale selection frames

Step 3.

- Fix the counting period of the timer, in the Timer Period Register (TBPRD) setting frame. Use one of the following features:
 - Write the hexadecimal value in the TIMER_PERIOD edit box, or
 - Drag the cursor of the scroll bar to set the period in microseconds

Step 4.

- Set the compare value, in the Compare A Register (CMPA) setting frame. Use one of the following features:
 - Write the value (in percents of timer period) in the COMP.VALUE edit box, or
 - Drag the cursor of the scroll bar to set the compare value in percents of timer period

Step 5.

- Click on the **Run** button to start the experiment

Step 6.

- Visualize the values in the Period interrupts count variable text box while the program is running

Step 7.

- Click on the **Stop** button to stop the experiment and visualize the values in the Period interrupts count variable text box and Compare interrupts count variable text box when the program has stopped

5.10.2.4 DSC software description

Figure 5.27 presents the flowchart of the DSC program that performs this application.

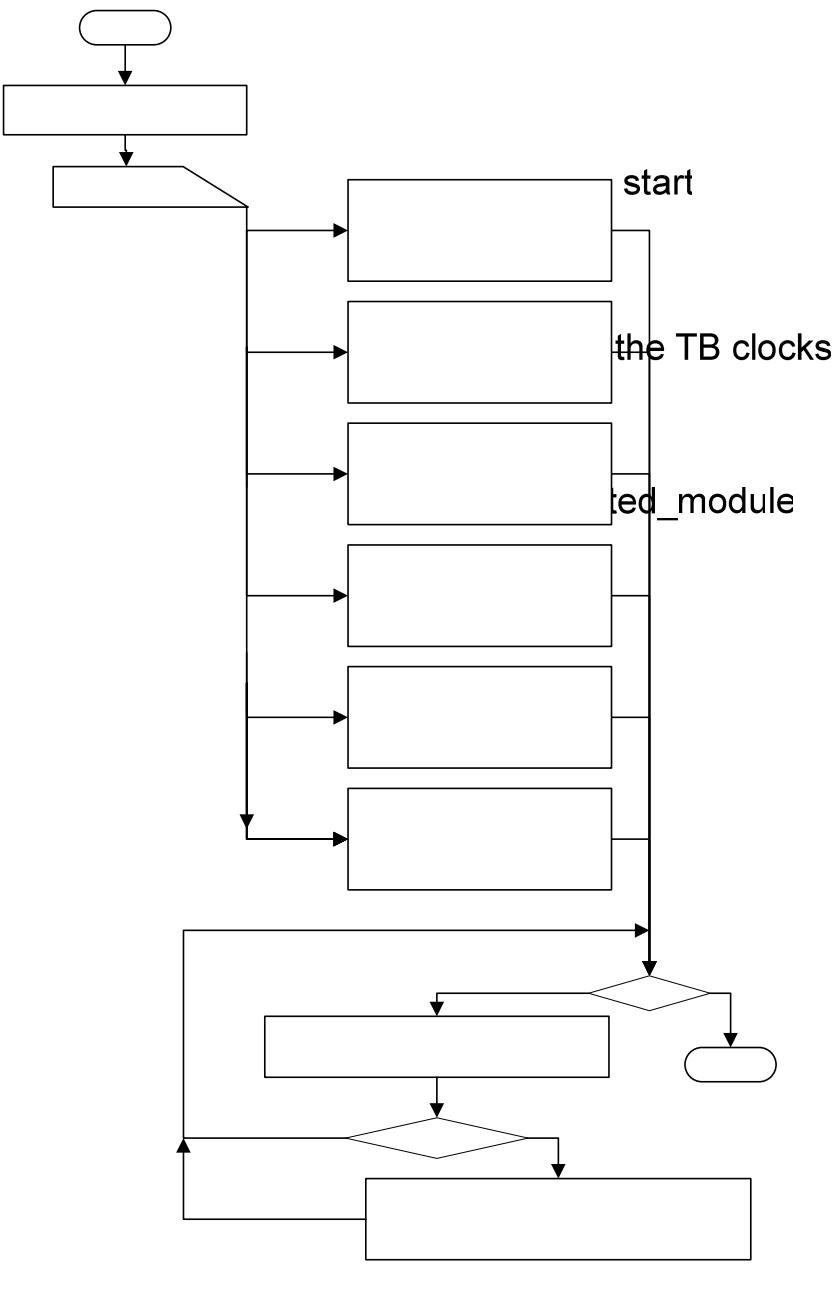


Figure 5.27.A. General Purpose Timer Application - Advanced Timer Programming DSC program flowchart

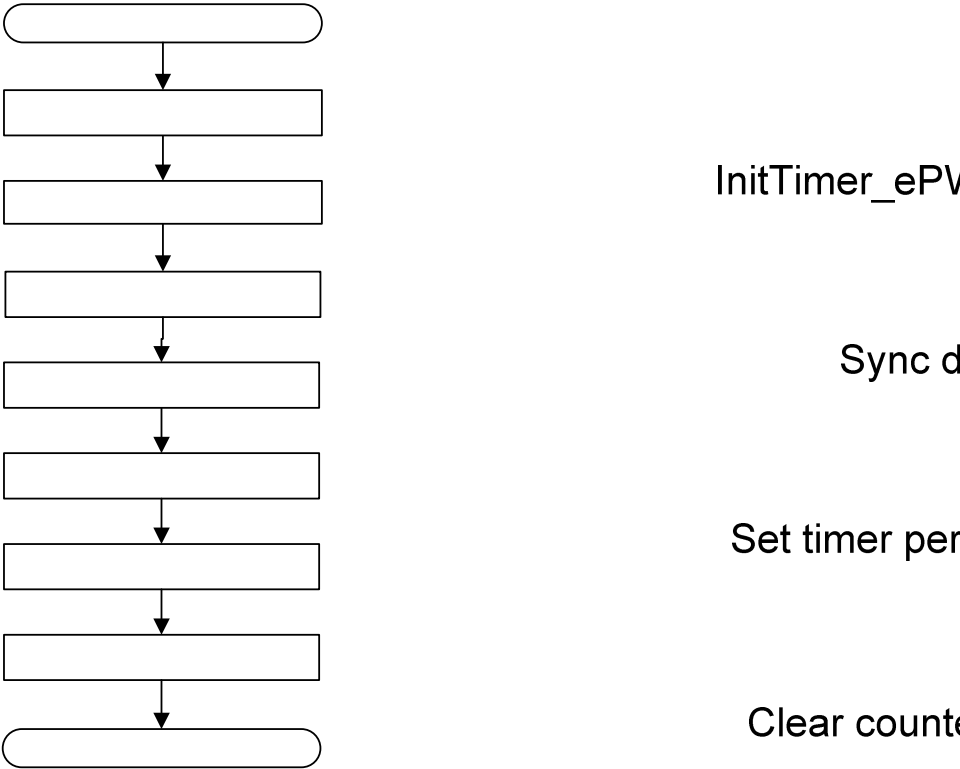


Figure 5.27.B. General Purpose Timer Application - Advanced Timer Programming DSC program flowchart

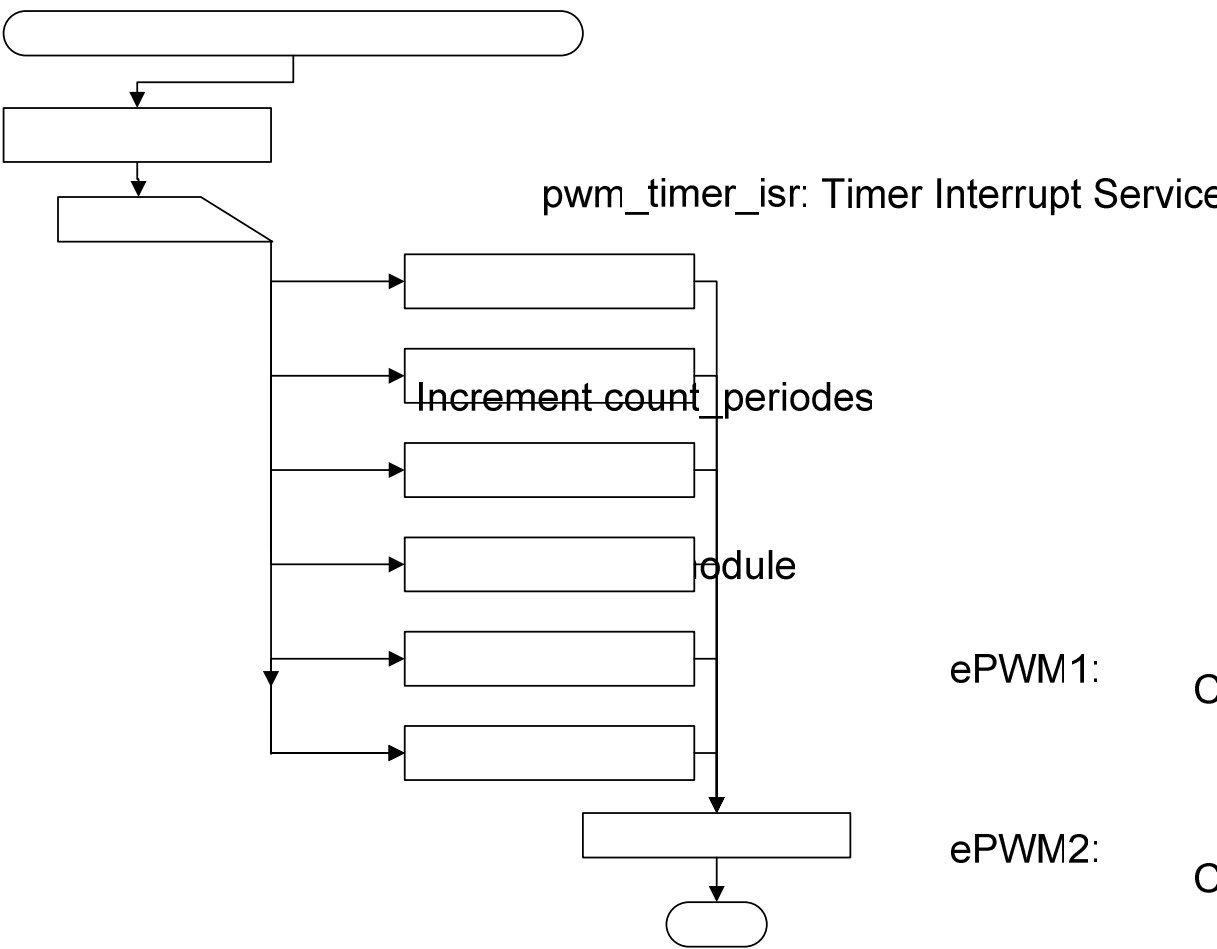


Figure 5.27.C. General Purpose Timer Application - Advanced Timer Programming DSC program flowchart

5.10.2.5 Program files

The file name containing the DSC program for this application is **gpt_2.c**. Activating the Show code action control button in the application dialog can inspect it. The C header file called **gpt_2.h** holds the variables and function declarations for the application.

5.10.2.6 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**, and one interrupt service routine functions: **pwm_timer_isr**. Program variables are allocated in external data memory starting at address **0x3F9000**:

- 0x3F9000: DemoGPT2.stop** program stop index (set to 1 to end the program)
- 0x3F9001: DemoGPT2.tbctl_value** variable containing the value for Time-Base Control Register (TBCTL)

0x3F9002: DemoGPT2.timer_period	variable containing the value for timer period register (TBPRD)
0x3F9003: DemoGPT2.compare_value	variable containing the value for Compare A register (CMPA)
0x3F9004: DemoGPT2.count_periodes	variable incremented each time a timer period interrupt is serviced
0x3F9005: DemoGPT2.count_compares	variable incremented each time a timer compare interrupt is serviced
0x3F9006: DemoGPT2.selected_module	variable specifying which ePWM module are to be used

5.10.2.7 Application results evaluation

The application can run under various timer modes (up-count, down-count, up-down-count and freeze counter mode). When the application is stopped, the final values of the count variables help understanding how a ePWM timer works.

The application DSC program illustrates a simple structure of interrupted code in C language.

5.11 Watchdog (WD) Module Application

5.11.1 Purpose

Experience the features offered by the Watchdog Module.

The application consists in activating ePWM1 timer period interrupt routine in which a counter variable is increased and the WD control registers are updated. If the timer interrupt period is larger than the watchdog period or the watchdog updated key values are wrong, the module generates a processor reset.

5.11.2 Screen Presentation

The dialog presented in Figure 5.28 is accessible by selecting the « **Watchdog** » button in the **Processor Evaluation Control Panel** menu.

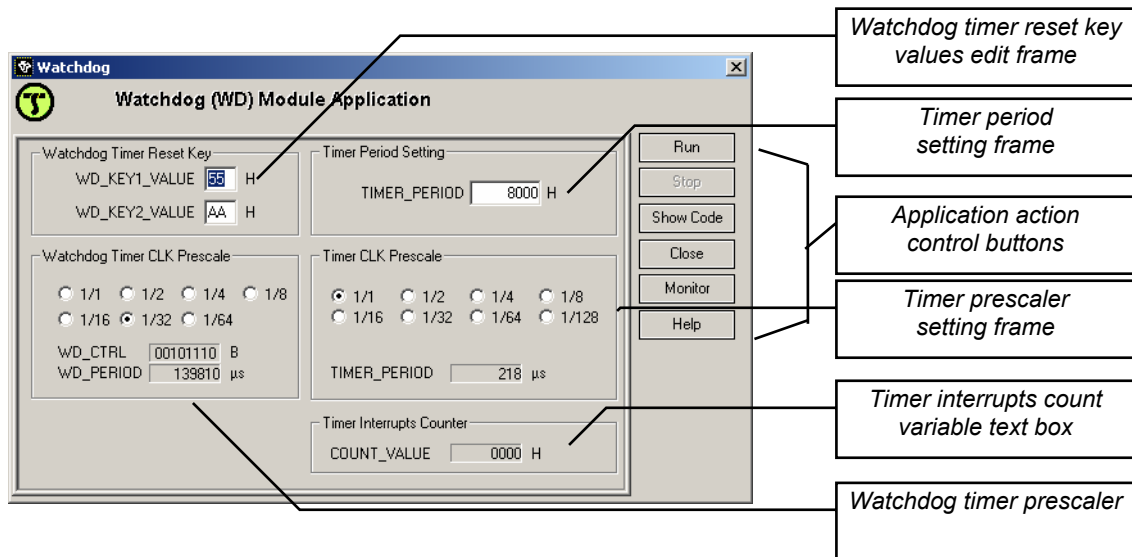


Figure 5.28. Watchdog (WD) Module Application dialog

5.11.3 Actions to run the application

Step 1.

- Click on the **Run** button to start the experiment

Step 2.

- Inspect the timer interrupt count variable text box to see how the COUNT_VALUE increases continuously while the program is running. The program runs endlessly as long as no reset condition occurs.

Step 3.

Do on-line modifications to generate reset conditions:

- modify one or both of the values of the keys written to the WD Reset Key Register using the edit boxes in the WD timer reset key values edit frame

or,

- in the WD timer period and timer period setting frames, select the periods for WD timer and ePWM1 timer, so that the ePWM1 timer period is greater than the WD timer period

Note that when a WD reset condition occurs, a RESET signal is generated to the processor and the application stops. To make the application run again, the reset condition must be eliminated by a proper setting of the WD key values (**55 H** followed by **AA H**) and WD and ePWM1 timer periods (ePWM1 timer period < WD timer period).

5.11.4 Program files

The file name containing the DSC program for this application is **wd.c**. Activating the Show code action control button in the application dialog can inspect it. The C header file called **wd.h** holds the variables and function declarations for the application.

5.11.5 DSC software description

Figure 5.29 presents the flowchart of the DSC program that performs this application.

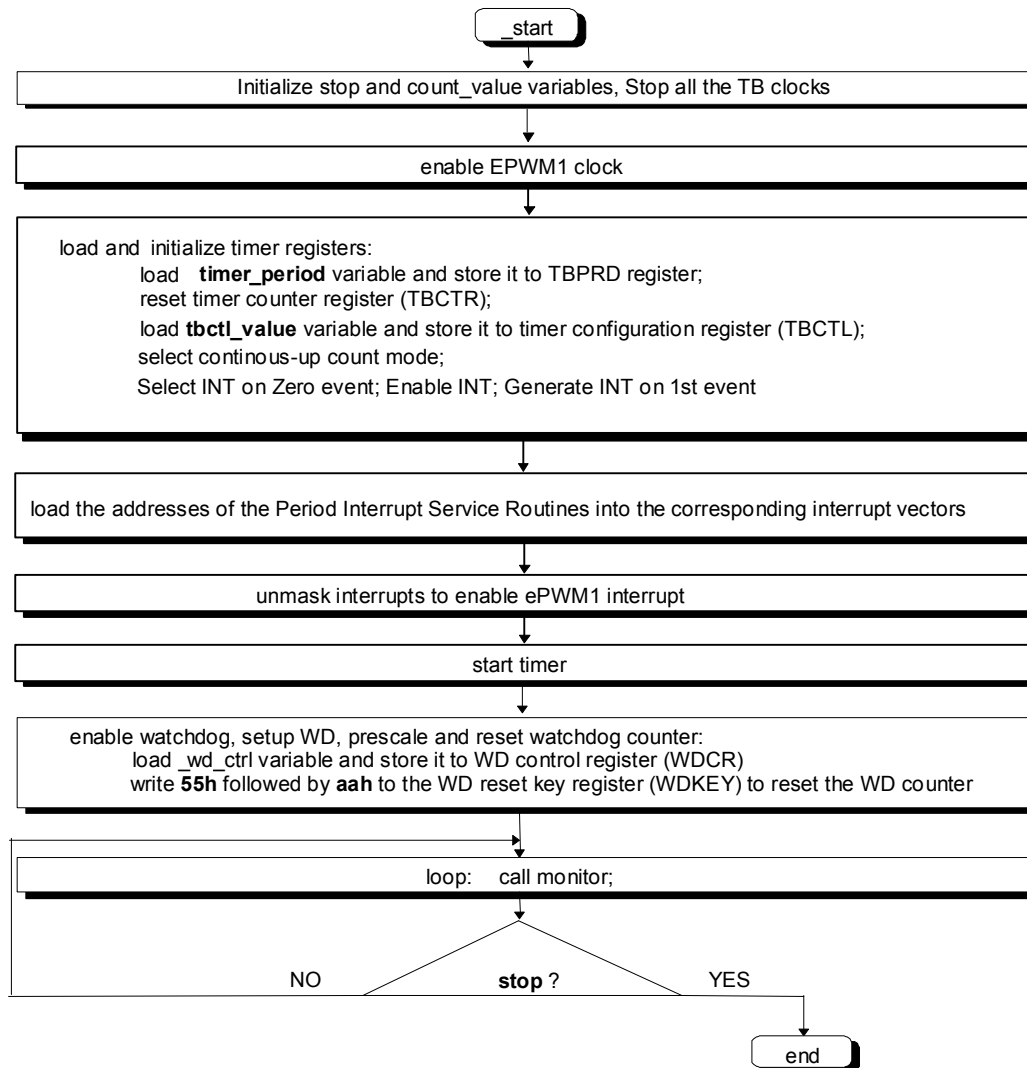


Figure 5.29.A. Watchdog (WD) Module Application DSC program flowchart

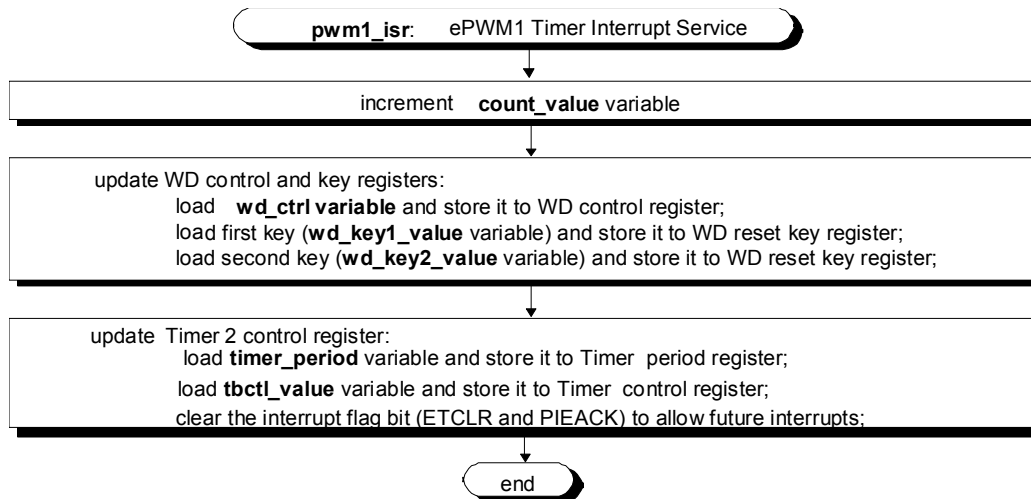


Figure 5.29.B. Watchdog (WD) Module Application DSC program flowchart

5.11.6 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**, and 1 interrupt service routine function: **pwm1_isr**.

Program variables are allocated in external data memory starting at address **0x3F9000**:

0x3F9000: DemoWD.stop	program stop index (set to 1 to end the program)
0x3F9001: DemoWD.wd_key1_value	variable containing the first value to write to WD Reset Key Register
0x3F9002: DemoWD.wd_key2_value	variable containing the second value to write to WD Reset Key Register
0x3F9003: DemoWD.wd_ctrl	variable containing the value for WD Control Register
0x3F9004: DemoWD.tbctl_value	variable containing the value for Time Base Control Register
0x3F9005: DemoWD.timer_period	variable containing the value for Timer Period Register
0x3F9006: DemoWD.count_value	variable incremented each time a Timer period interrupt is serviced

5.11.7 Application results evaluation

The experiment helps understanding the benefits of a watchdog module in an application in which a real time process is controlled. If the DSC program runs under normal correct conditions, the WD counter is reset each moment the real time interrupt is serviced by successively writing two well defined values to the WD key register.

If by various reasons, the DSC program has lost the control of the processor, a WD reset condition occurs and the WD resets the processor. In this way the application stops because it can no longer control correctly the real time process.

5.12 Inter-Integrated Circuit (I2C) Module Application

5.12.1 Purpose

Send/receive user specified characters codes to/from the high-speed on-chip Inter-Integrated Circuit Interface.

The I2C is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit-transfer rate. The I2C is normally used for communication between the DSC controller and external peripherals or another controller.

This application programs the I2C to send/receive a 16-bit stream representing a code entered by you in the application dialog. External connection is needed to route the I2C output pin back to the I2C input pin or, as an alternative, to connect the interface to a similar I2C interface of another controller.

5.12.2 Screen Presentation

The dialog presented in Figure 5.30 is accessible by selecting the « I2C » button in the **Processor Evaluation Control Panel** menu.

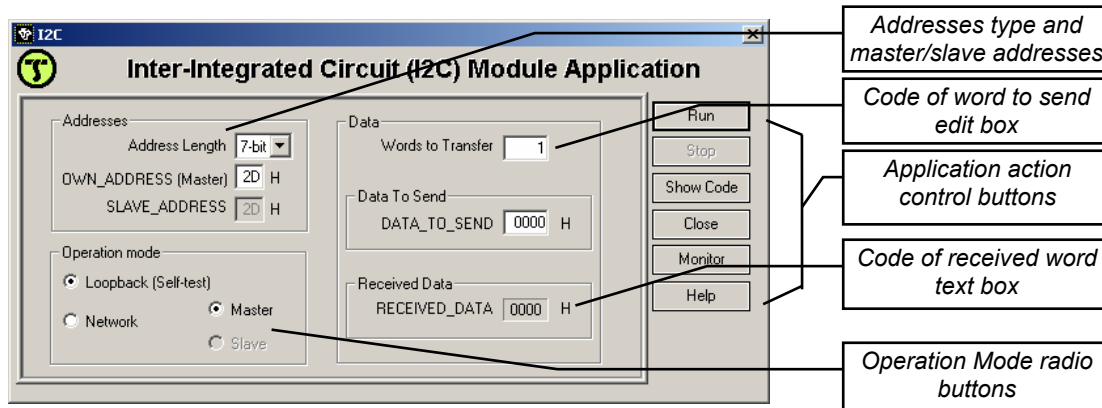


Figure 5.30. Inter-Integrated Circuit (I2C) Module Application dialog

5.12.3 Actions to run the application

Step 1.

- Select the **Operation mode**, **Loopback** mode to connect internally the reception to transmission of I2C or **Network** if external I2C device is connected.

Step 2.

- Select the **Master** or **Slave** mode. Available only in **Network** mode.

Step 3.

- Enter a code composed of two hexadecimal values in the OWN_ADDRESS edit box.

Step 4.

- Enter a code composed of two hexadecimal values in the SLAVE_ADDRESS edit box.

Step 5.

- Enter a code composed of four hexadecimal values in the DATA_TO_SEND edit box.

Step 6.

- Click on the **Run** button to start the experiment.

Step 7.

- If the **Loopback** mode was selected the 16-bit length data, representing the code to send, will be echoed by I2C and will appear in the REC_DATA text box.

5.12.4 DSC software description

Figure 5.21 presents the flowchart of the DSC program that performs this application.

5.12.5 Program files

The file name containing the DSC program for this application is **i2c.c**. Activating the Show code action control button in the application dialog can inspect it. The header file called **spi.h** holds the variables and function declarations for the application.

5.12.6 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**, and two routine functions: **send** and **receive**.

Program variables are allocated in external data memory starting at address **0x3F9000**:

0x3F9000: Demol2C.stop	program stop index (set to 1 to end the program)
0x3F9001: Demol2C.data_to_send	16 bit value representing the code of a word to be sent on the I2C serial line
0x3F9002: Demol2C.rec_data	16 bit value representing the code of a word received on the I2C serial line
0x3F9003: Demol2C.own_address	16 bit value representing the master address on the I2C
0x3F9004: Demol2C.slave_address	16 bit value representing the slave address on the I2C
0x3F9005: Demol2C.operation_mode	16 bit value representing the flag for selection of operating mode
0x3F9006: Demol2C.master	16 bit value representing the flag for selection of master / slave mode

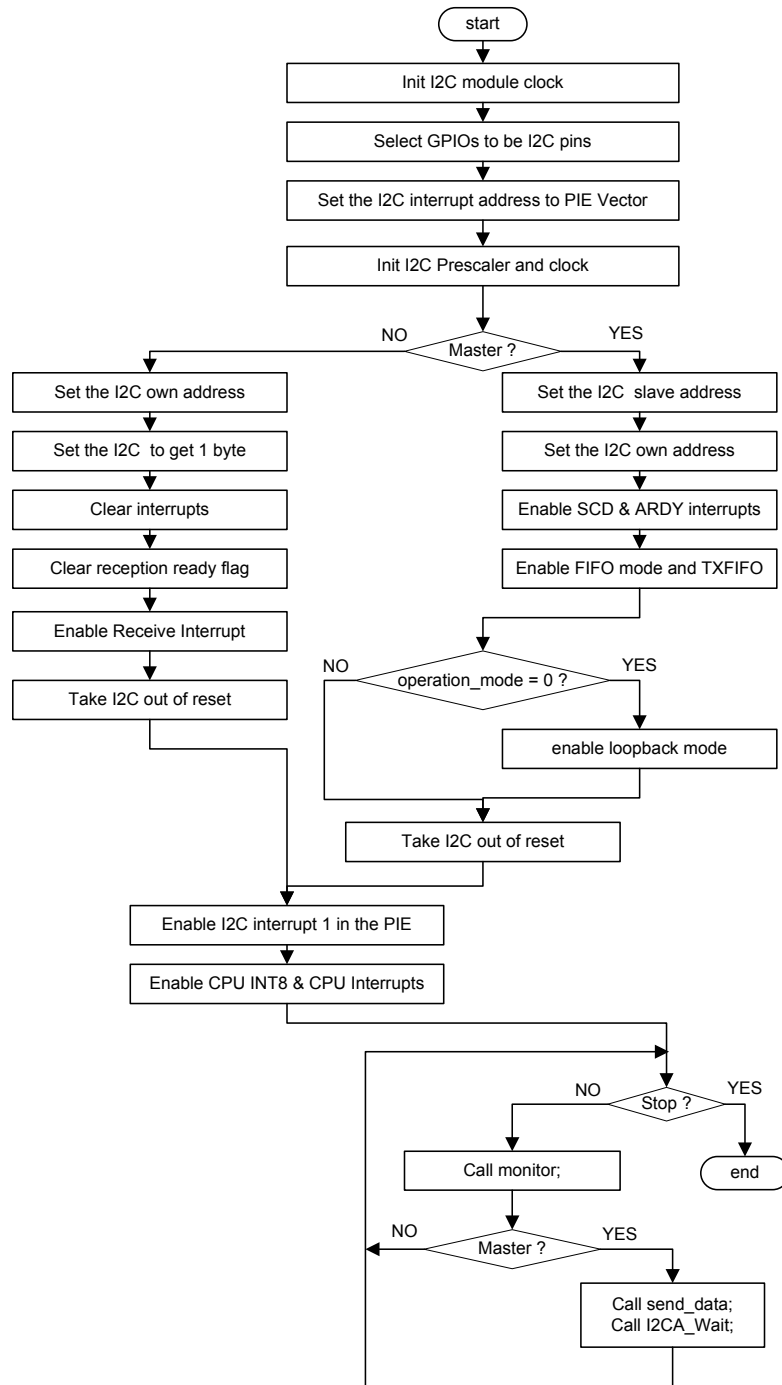


Figure 5.31.A. Inter-Integrated Circuit (I2C) Module Application DSC program flowchart

5.12.7 Application results evaluation

The Inter-Integrated Circuit is programmed in this application to continuously transmit/receive 16 bit streams on the input/output serial lines. The transmitted values are entered by you in the application dialog. In Loopback mode, received codes are then identical with the transmitted ones. The application dialog displays also the code received on the I2C serial input line. Inspecting DSC program code helps understanding the programming of the on-chip Inter-Integrated Circuit Module.

5.13 Multichannel Buffered Serial Port (McBSP) Module Application

5.13.1 Purpose

Send/receive user specified characters codes to/from the high-speed on-chip Multichannel Buffered Serial Port Interface.

The McBSP is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (8 / 12 / 16 / 20 / 24 / 32 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The McBSP is normally used for communication between the DSC controller and external peripherals or another controller.

This application programs the McBSP to send/receive a data stream, with specified length, representing a code entered by you in the application dialog. In **Network** mode, external connection is needed to route the McBSP output pin back to the McBSP input pin or, as an alternative, to connect the interface to a similar McBSP interface of another controller.

5.13.2 Screen Presentation

The dialog presented in Figure 5.32 is accessible by selecting the « **McBSP** » button in the **Processor Evaluation Control Panel** menu.

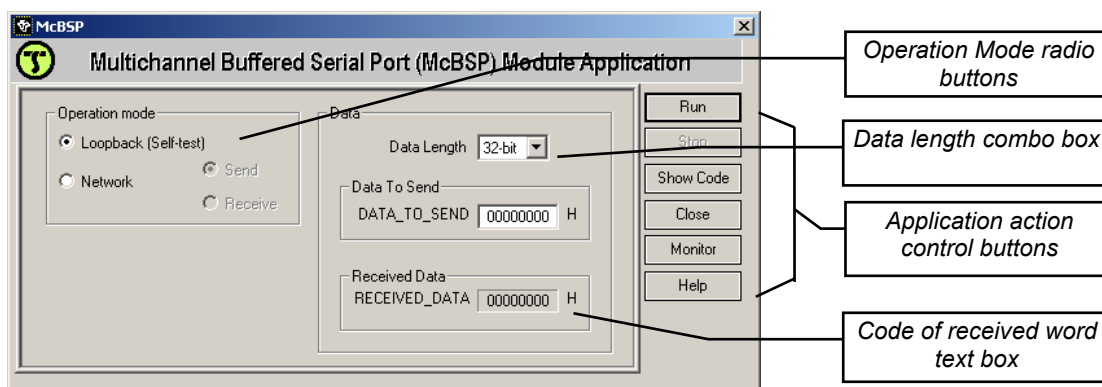


Figure 5.32. Multichannel Buffered Serial Port (McBSP) Module Application dialog

5.13.3 Actions to run the application

Step 1.

- Select the **Operation mode**, **Loopback** mode to connect internally the reception to transmission of McBSP or **Network** if external McBSP device is connected.

Step 2.

- Select the **Send** or **Receive** mode. Available only in **Network** mode.

Step 3.

- Select the **Data Length**.

Step 4.

- Enter a code composed of four hexadecimal values in the DATA_TO_SEND edit box.

Step 5.

- Click on the **Run** button to start the experiment.

Step 6.

- If the **Loopback** mode was selected the data, representing the code to send, will be echoed by McBSP and will appear in the REC_DATA text box.

5.13.4 DSC software description

Figure 5.21 presents the flowchart of the DSC program that performs this application.

5.13.5 Program files

The file name containing the DSC program for this application is **mcbbsp.c**. Activating the Show code action control button in the application dialog can inspect it. The header file called **mcbbsp.h** holds the variables and function declarations for the application.

5.13.6 Variables and functions

The program consists of a main function with the entry point at internal program memory address **0x3F8080**, and two routine functions: **send** and **receive**.

Program variables are allocated in external data memory starting at address **0x3F9000**:

0x3F9000: DemoMcBSP.stop	program stop index (set to 1 to end the program)
0x3F9001: DemoMcBSP.data_length	16 bit value representing the data length on the McBSP
0x3F9002: DemoMcBSP.data_to_send	16 bit value representing the code of a word to be sent on the McBSP serial line
0x3F9004: DemoMcBSP.rec_data	16 bit value representing the code of a word received on the McBSP serial line
0x3F9006: DemoMcBSP.operation_mode	16 bit value representing the flag for selection of operating mode
0x3F9007: DemoMcBSP.master	16 bit value representing the flag for selection of master / slave mode

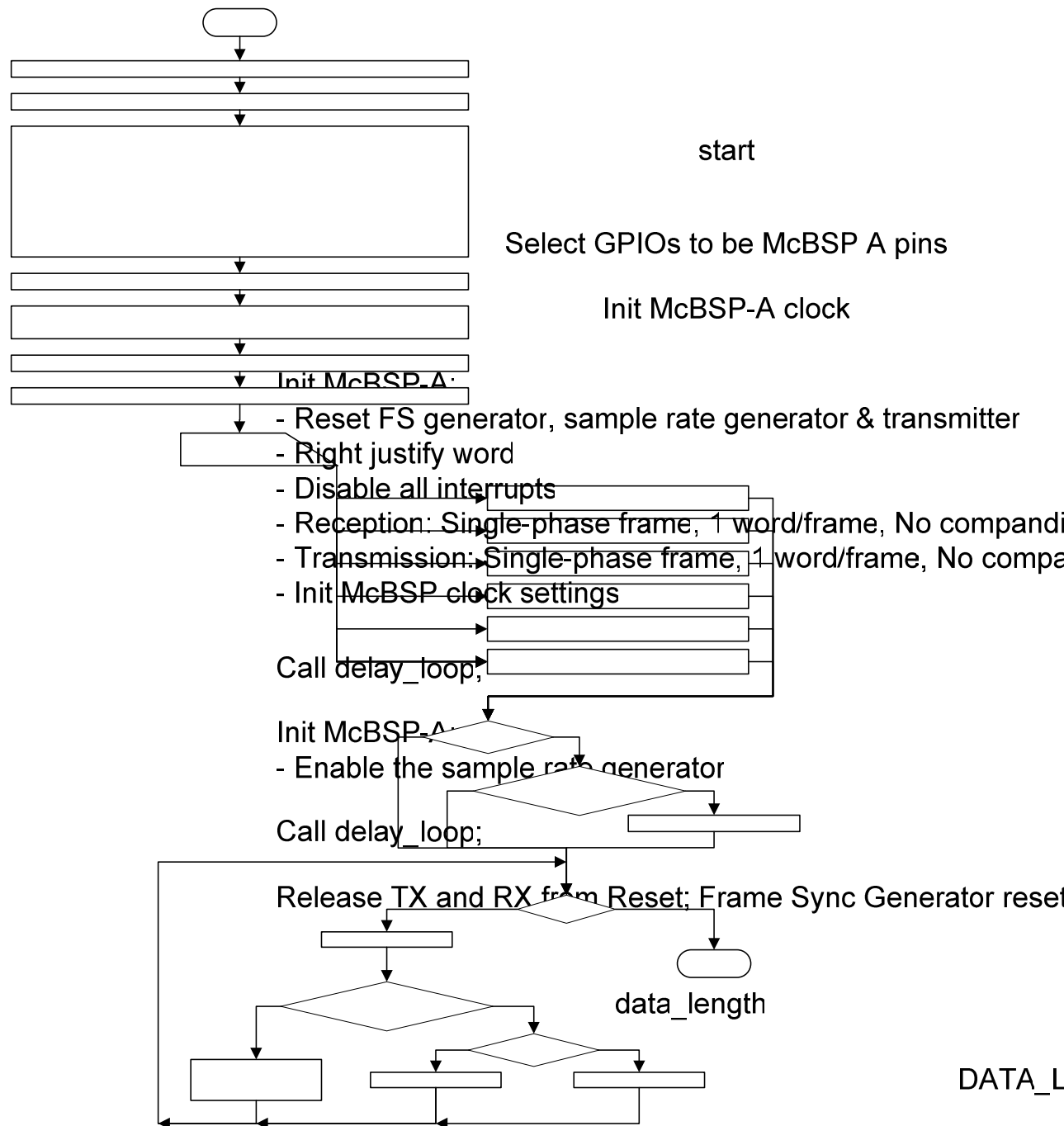


Figure 5.33.A. Multichannel Buffered Serial Port (McBSP) Module Application DSC program flowchart

DATA_L

DATA_L

DATA_L

DATA_L

DATA_L

DATA_L

NO

5.13.7 Application results evaluation

The Multichannel Buffered Serial Port is programmed in this application to continuously transmit/receive 8 / 12 / 16 / 20 / 24 / 32 bit streams on the input/output serial lines. The transmitted values are entered by you in the application dialog. In Loopback mode, received codes are then identical with the transmitted ones. The application dialog displays also the code received on the McBSP serial input line. Inspecting DSC program code helps understanding the programming of the on-chip Multichannel Buffered Serial Port Module.

6 Motion Control Graphical Analysis Tool -

DSPMOT (*MCK28335 kits*)

This chapter describes the structure and functions of the program module **DSPMOT**, an advanced graphical tool for the analysis and evaluation of motion control applications implemented on the **MSK28335 DSC board**.

Contents

6.1. Basic principles of DSPMOT applications

6.2. DSPMOT menu commands

6.3. Advanced DSPMOT features

6.1 Basic principles of DSPMOT applications

The **DSPMOT** program represents an advanced tool for digital motion control programs analysis and evaluation.

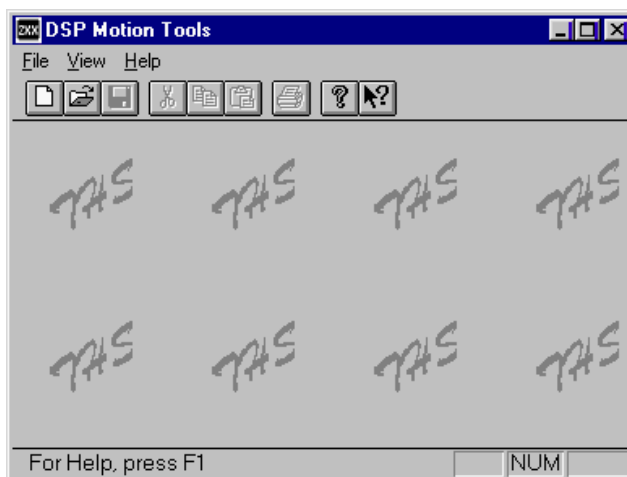
The DSPMOT is included in the MCK28335 kits and can be opened using:

- For **MSK28335 kit** : "**Start | Programs | MCWIN28335 | DSPMOT32**"
- For **MSK28335 kit Pro**: "**Start | Programs | MCWIN28335 Professional | DSPMOT32**".

The program allows you to download applications to the **MSK28335 DSC board**, examine/set the value of program variables, execute the DSC program. By calling from your program appropriate functions (supplied with the package), you may activate communication, data logging and linear reference functions in his application. Thus, all the powerful Windows-based graphical features of the **DSPMOT** program may be used, and advanced debugging of the program may be performed, significantly reducing the development time of the motion control applications.



The **DSPMOT** program represents a work platform for the analysis and evaluation of motion control applications executed on the **MSK28335 DSC board**. Basically, you may define a specific working environment for a given motion application. This environment consists on all the parameters associated to that application, as path and file names of TMS320F28335 executable programs, variables addresses and initial values, as well as graphical settings, specific data logger information, reference generator and digital controllers parameters. All the working context of a given application may be saved on specific environment description data files. These binary files, having the default extension **.MOT**, may be loaded and used at latter time in order to work with the associated application files and environment.



Usually, at the start of **DSPMOT**, the initial screen of the program will be opened, as in Figure 6.1.

Using the **File | New** menu command or toolbar icon (see par.6.2.1 for details), you may start to define a new project environment. An extended menu replaces the initial opening menu from Figure 6.1, with more other menu commands (see par. 6.2 for the [detailed menu commands description](#)). You will be able to attach to the newly defined project a specific DSC executable code file (a ***.out** and its associated ***.map** file).

Figure 6.1 Opening window of the **DSPMOT** program

At a basic level of **DSPMOT** use, for any TMS320F28335 application which may be loaded into the external RAM memory of the **MSK28335 DSC board** for debugging purposes, you may examine, modify and/or select specific variables to be initialized for that application, and execute that application from **DSPMOT** (see the **Motion** menu command description in par. 6.2.3).

At an advanced level of **DSPMOT** use, if specific function calls are included in your application, supplementary powerful features may be added in order to perform its debugging.

Thus, it is possible to include the communication features of the **MON28335** monitor program into your application, allowing on-line watching/modification of program variables, during the motion execution. See par. 6.2.3 for the description of the setup and watch variables procedures, and par. 6.3.1 for details about the built-in communication inclusion in your application.

If the data logger module is also included in your application, up to 8 program variables may be stored simultaneously during the real-time execution of the motion, and then up-loaded and visualized in the **DSPMOT** graphical environment. See par. 6.2.2 for the description of all the graphical features of **DSPMOT**, and par. 6.3.2 for details about the built-in data logger inclusion in your application.

If the reference generator module is included in your application, you may define the motion reference at a high level, in **DSPMOT**, download it and execute it automatically on the **MSK28335 DSC board**. See par. 6.2.3 for the description of the reference generator of **DSPMOT**, and par. 6.3.3 for details about the built-in reference generator inclusion in your application.

Finally, if the control module is included in your application, you may define the controllers parameters at a high level in **DSPMOT**, download them and use them on the **MSK28335 DSC board**. See par. 6.2.3 for the description of the controllers parameters settings in **DSPMOT**, and par. 6.3.4 for details about the built-in controllers inclusion in your application.

As the **DSPMOT** program is an MDI application, multiple projects may be opened simultaneously, allowing you to test and compare different motion solutions for a given motion structure.

6.2 DSPMOT menu commands

Several menu commands and associated toolbar icons may be used in **DSPMOT** in order to activate its different functions. The following sub-paragraphs give a complete description of the menu and sub-menu commands of **DSPMOT**.

6.2.1 File menu commands

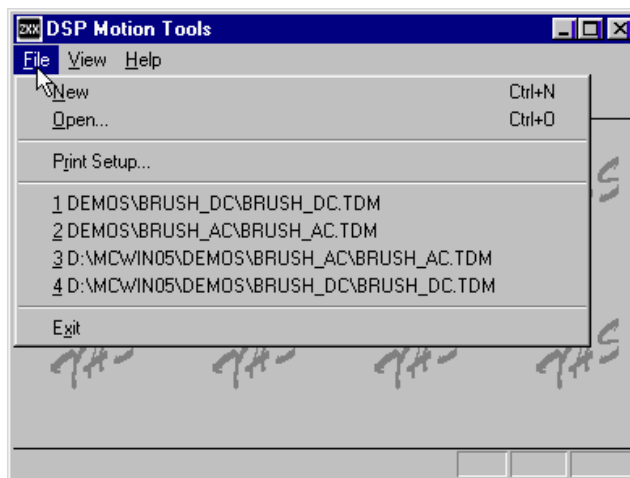


Figure 6.2. File menu command at start of DSPMOT

At the start of the **DSPMOT** program, the **File** menu command has the following menu sub-commands:

- **File | New**
- **File | Open**
- **File | Print Setup**
- **File | Exit**

It also contains the list of the up to four last opened *.MOT projects.

Figure 6.2 presents the **File** menu command after the start of the **DSPMOT** program.

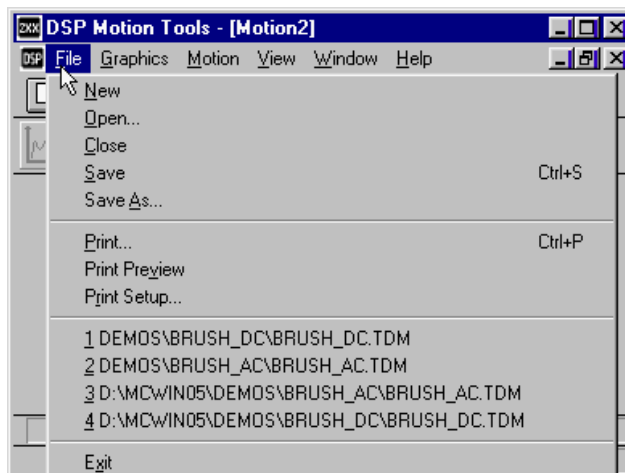


Figure 6.3. File menu command when at least one *.MOT project is opened

If at least one motion project is opened in the **DSPMOT** program, the **File** menu command has the following menu sub-commands (see Figure. 6.3):

- **File | New**
- **File | Open**
- **File | Close**
- **File | Save**
- **File | Save As...**
- **File | Print...**
- **File | Print Preview**
- **File | Print Setup**
- **File | Exit**

It also contains the list of the up to four last opened *.MOT projects.

Hot-key : [Alt-F]

The following paragraphs give the complete description of each of these menu sub-

commands.

6.2.1.1 File | New menu command

The **File | New** menu command allows you to create a new project. When selected, this command will start a new motion project. It automatically names it as **Motion1**, **2**, etc. Both program menu and menu toolbar will change, according to the commands which may be given by you in the framework of a motion project.

After starting a new project, you need to define the associated DSC application name and location (the *.out and *.map files), using the **Motion | Setup DSC Variables**, « **Load From *.map file** » (see par. 6.2.3 for details). Using the same command it is also possible to select the list of variables to be logged during the motion. Several **Graphics** menu commands allow the setting of graphical parameters as subplots contents, colors, line styles and patterns, etc. (see par. 6.2.2 for details).

A project may be saved at any time. A newly created project will automatically prompt you to save it when closing, similar to the **File | Save As...** menu command (see par. 6.2.1.4). A *.MOT file is created and may be loaded at a latter time, using the **File | Open** menu command (see par. 6.2.1.2 for details).

Hot-key : [Alt-F N]

Standard toolbar icon button :



6.2.1.2 File | Open menu command

The **File | Open** menu command allows you to open a previously saved motion project. When selected, this command will open the standard file open dialog of Windows, allowing the search and selection of the project file. By default, the listed files are the standard **DSPMOT** project files, having the specific **MOT** file suffix. Once a file is selected, it is opened, and the project parameters and settings are loaded in the **DSPMOT** environment. You may then proceed to the analysis and development of his application.

Important remark : If the DSC application has not changed since the last saving of the associated *.MOT project, all the settings (related to the motion as well as to the graphics parameters) are valid and may be used directly (i.e. you may directly load and execute the motion application).

On the contrary, **IF THE DSC APPLICATION HAS CHANGED** (i.e. if new *.out, *.map files were generated following a new compilation and/or link of the motion application, or if the motion project location - path - has changed), you **MUST RELOAD** the associated *.map and *.out files (use first the **Motion | Set DSC Variables** « **Load From *.map file** » command to reload the *.map file and store the path of the DSC project files - see par. 6.2.3 for details). It is recommended in this case to save the project, using the **File | Save** menu command.

After loading a project, if any of the settings are changed, you must save the project in order to keep all the changes valid for future use. Use **File | Save** menu command to overwrite the previous project *.MOT file, or the **File | Save As...** menu command to save the project on a new *.MOT file (see par. 6.2.1.4 for details).

Hot-key : [Alt-F O]

Standard toolbar icon button :



6.2.1.3 File | Close menu command

The **File | Close** menu command allows you to close a motion project. If the project has changed since the last saving, the program will prompt you if wanting to save the project before closing. Answering «**Yes**» will

overwrite the previous project **MOT** file. Another option is to cancel the closing procedure, and to use the **File | Save As...** menu command to save the project under a new project name.

Hot-key : [Alt-F C]

6.2.1.4 File | Save and File | Save As... menu commands

The **File | Save** and **File Save As...** menu commands allow you to save the active motion project context (i.e. DSC executable file name and location, variables selections, graphical settings, reference and controllers settings, etc.).

For newly created projects (named by default as **Motion1, 2**, etc.), this command will automatically open the standard Windows **File Save As... dialog**, allowing you to choose the name and the path of the motion project file and to save it. By default, the saved files are the standard **DSPMOT** project files, having the specific **MOT** file suffix.

Projects which were already saved before and have a specific name will be directly saved under the same name, without prompting you. Use the **File | Save As...** menu command instead, if wanting to save a modified project under a different name, in order to keep unmodified its previous version.

A saved motion project may be reloaded anytime latter. A load operation (use the **File | Open** menu command for this purpose) will completely restore the saved motion project context, allowing you to continue your work from the same status as that one at the save moment.

File Save

Hot-keys : [Ctrl-S] or [Alt-F S]



Standard toolbar icon button :

File Save As...

Hot-key : [Alt-F A]

6.2.1.5 File | Print menu command

The **File | Print** menu command allows you to print the graphical window of a motion project. The program will open the standard Windows **Print dialog**, allowing you to set the printing parameters (as printer, pages to print, number of copies, etc.).

Hot-key : [Ctrl P] or [Alt-F P]



Standard toolbar icon button :

6.2.1.6 File | Print Preview menu command

The **File | Print Preview** menu command allows you to preview the graphical window of a motion project before printing. The program will open the standard Windows **Print Preview dialog**, allowing you to preview the page contents as it will appear on the printed page, as well as to print it, eventually from this dialog, instead to use the **File | Print** menu command.

Hot-key : [Alt-F V]

6.2.1.7 File | Print Setup menu command

The **File | Print Setup** menu command allows you to setup the current printer parameters. The program will open the standard Windows **Print Setup dialog**, allowing you to select the printer and set its parameters.

Remark: The print settings will be valid and remain active for the current session of **DSPMOT** program. They apply only for print operations generated from the **DSPMOT** program, and do not change the default system settings as defined in the Windows environment.

Hot-key : [Alt-F R]

6.2.1.8 File | Exit menu command

The **File | Exit** menu command allows you to exit **DSPMOT**. If one or more currently opened projects were modified since their last saving, the program prompts you for each project to specify if wanting to save it or not before closing the program.

When selecting the save option for newly created projects (named by default as **Motion1**, **2**, etc.), the program will automatically open the standard Windows **File Save As...** dialog, allowing you to choose the name and the path of the motion project file and to save it. By default, the saved files are the standard **DSPMOT** project files, having the specific **MOT** file suffix.

When selecting the save option for the projects which were already saved before and have a specific name, the project will be directly saved under the same name, without prompting you. Cancel the exit procedure and use the **File | Save As...** menu command instead, if wanting to save a modified project under a different name, in order to keep unmodified its previous version.

6.2.2 Graphics menu commands

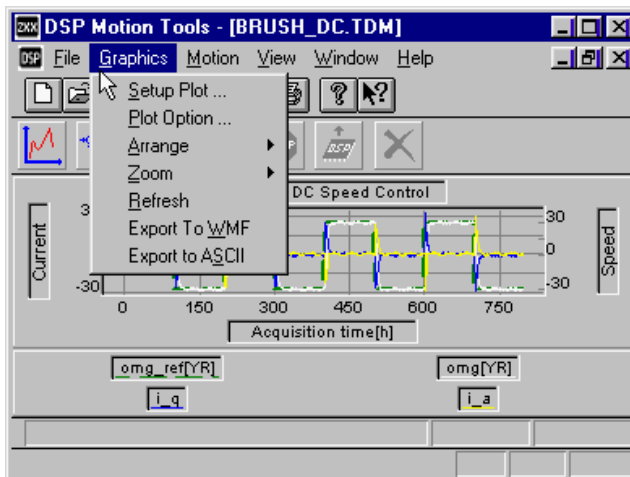


Figure 6.4. Graphics menu command

If at least one motion project is opened in the **DSPMOT** program, the **Graphics** menu command has the following menu sub-commands (see Figure 6.4):

- **Graphics | Setup Plot...**
- **Graphics | Plot Option...**
- **Graphics | Arrange**
- **Graphics | Zoom**
- **Graphics | Refresh**
- **Graphics | Export to WMF**
- **Graphics | Export to ASCII**

Depending on the state of the motion project, some of these menu sub-commands will be enabled or not, allowing you to execute only the allowed operations for a given project environment.

Hot-key : [Alt-G]

Important Notice : The same menu command may be opened by a **right-button mouse click** performed by you when the mouse cursor is located somewhere in the active area of the graphic region of the **DSPMOT** window.

The following paragraphs give the complete description of each of these menu sub-commands.

6.2.2.1 Graphics | Setup Plot menu command



The **Graphics | Setup Plot** menu command allows you to select and group on specific graphic subplots the motion project variables which will be stored during the motion execution through the data logging procedure (see par. 6.3.2 for details about the data logging feature). Up to four subplots may be defined by you for a given project.

The list of variables which will be stored during the data logging must be defined using the **Motion | Setup DSC Variables** menu command (see par. 6.2.3 for details). Once this list defined, you may accede the **Graphics | Setup Plot dialog** in order to select the corresponding variables and distribute them on the graphics subplots for visualization. Figure 6.5 presents the dialog window opened in this case, for a given motion project.


As one can see from the Figure 6.5, the dialog contains the complete list of the curves selected to be stored, in the top of it. Up to four different subplots may be defined. For each subplot, the curves to be plotted on it may be chosen from the complete list of stored variables, grouped under the title « **Available Curves** ».


You may switch between the subplots using the corresponding tabs associated to each subplot. By default, each subplot tab is named as « **Subplot options** ». You may freely define each of the subplot names. The « **Subplot Title** » edit box contains the actual (if it was defined) subplot title. You may define or modify it at any time by editing this edit control.



Each subplot has an associated list of the selected curves to be displayed on that subplot, grouped under the name of « **Subplot Axis** ». The list may be updated by you by adding to / removing from it curves from the « **Available Curves** » list.

A variable may be added to the subplot curves list by selecting it in the « **Available Curves** » list, with a left-button mouse click (the selected variable becomes outlined - *omg_ref* in Figure 6.5, for example), and by pressing on one of the « **Add to the list** » buttons: press  to add it as the Y axis curve, or press the  button to add it as the X axis curve.


When adding a new variable to the subplot curves list, by having at the same time a selected (outlined) curve in the « **Subplot Axis** » list, the added variable will automatically replace the previous one from the « **Subplot Axis** » list, corresponding to the Y or X axis selected to be replaced.

A curve may be removed from the subplot curves list by selecting it in the « **Subplot Axis** » list, with a right-button mouse click (the selected curve becomes outlined), and by pressing the « **Remove from the list** » button .

Always, the first variable in the « **Available Curves** » list is the *Acquisition Time* variable. Usually, you will select some other variable to be added to the « **Subplot Axis** » list. When the first variable is selected and added to that list as a Y axis curve, the program automatically inserts by default, as the X-axis, the *Acquisition Time* variable .

The variables may be related to the left or to the right vertical axis of the subplot. Usually, the variables are introduced as related to the left vertical axis . If you want to change this setting to the right vertical axis, you need to double-click the vertical axis symbol, which will commute to the right vertical axis symbol . (A similar double-click on this symbol will reverse again the vertical axis to the left one).

If you want to use a special X-axis coordinate, different that the time variable (in order to visualize the dependence between two variables), you must select the desired X-axis variable in the « **Available**

Curves » list, and add it to the **« Subplot Axis »** list using the Add to X axis button . The newly selected variable will replace the time variable as the X-axis coordinate. When doing this, also the **« Monotonous X Axis »** checkbox control must usually be unchecked. This must be done in order to avoid erroneous plotting of data when the X-axis variable is not always increasing when representing the Y axis variables. For all the usual cases (including the standard *Acquisition Time* variable as the X-axis variable) check the **« Monotonous X Axis »** checkbox control in order to increase the plotting speed on the graphic window.

By default, the vertical axes do not have a name. You may freely define for each of the subplots names for the two vertical axes. The **« Labels »** group contains the actual (if it was defined) subplot left axis and right axis names. You may define or modify them at any time by editing the corresponding edit control.

Use the **« Apply »** button to effectively apply the defined settings and see their effect on the graphical window of **DSPMOT**, without closing the **Graphics | Setup Plot dialog**.

Use the **« OK »** button to effectively apply the defined settings and exit back to the graphical window of **DSPMOT**, by closing the **Graphics | Setup Plot dialog**.

Use the **« Cancel »** button to cancel all the defined settings and exit back to the graphical window of **DSPMOT**, by closing the **Graphics | setup Plot dialog**.

Use the **« Help »** button to get on-line information about the functions and the use of the **Graphics | Setup Plot dialog**.

Hot-key : [Alt-G S]

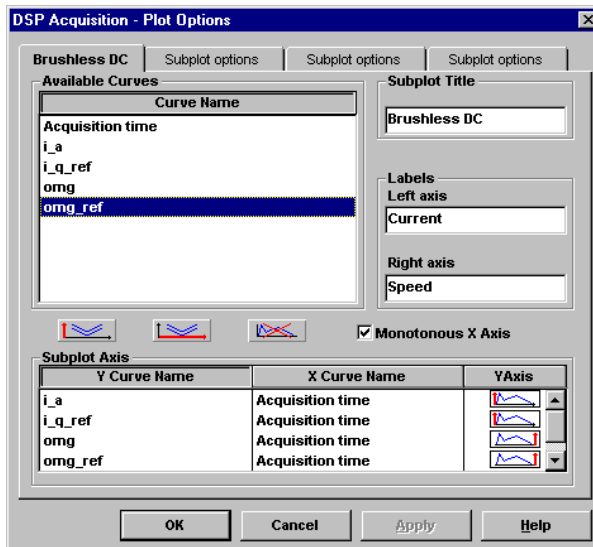


Figure 6.5. Graphics | Setup Plot dialog

6.2.2.2 Graphics | Plot Option menu command

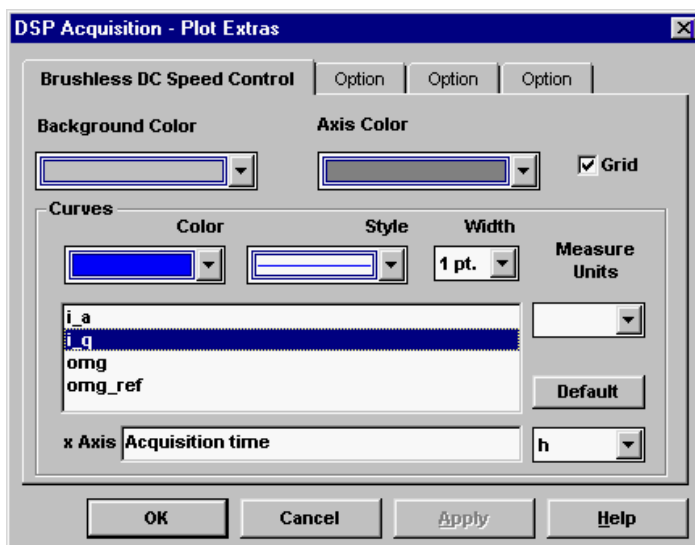


Figure 6.6. Graphics | Plot Option dialog

The **Graphics | Plot Option** menu command allows you to set the graphical parameters of all the variables selected to be plotted in any of the four subplots of **DSPMOT**, as colors, line width and pattern, background and axes colors, grid options and measurement units for each axis.

The list of variables which will be stored during the data logging must be defined using the **Motion | Setup DSC Variables** menu command (see par. 6.2.3 for details). Once this list defined, you may accede the **Graphics | Setup Plot dialog** in order to select the corresponding variables and distribute them on the graphics subplots for further visualization (see par. 6.2.2.1 for details).

With the plot variables selected in the **Graphics | Setup Plot dialog**, you may accede the **Graphics | Plot Option dialog** in order to examine / modify the predefined graphic attributes associated to the curves, axes, etc. Figure 6.6 shows the **Graphics | Plot Option dialog** opened in this case.

As one can see from Figure 6.6, the dialog contains the complete list of the variables selected to be stored for each of the possible four subplots which may be defined. For each subplot, any variable to be plotted on it may be selected from the list grouped under the title « **Curves** ».

You may switch between the subplots using the corresponding tabs associated to each subplot. By default, each subplot tab is named as « **Option** » if no name was given to the curve. Otherwise, if that name was defined, it is used as the tab name. (You may freely define each of the subplot names in the **Graphics | Setup Plot dialog**).

For each subplot, you may select any of the variables from the « **Curves** » list. Once a variable is selected (outlined) in the list, its graphical attributes are displayed and may be examined and/or modified by you.

Thus, one may modify :

- the curve color, using the « **Color** » drop-down list of available colors (up to 16 colors may be used);
- the curve style, using the « **Style** » drop-down list of available line styles (up to 5 line styles may be used);
- the curve width, using the « **Width** » drop-down list of available line widths (up to 3 line widths may be used)

You may also define/modify the curve measurement units (for display purposes only), which will be attached to the associated vertical axis for that curve. Use the « **Measure Units** » drop-down list to select a pre-defined measurement unit.

The background color may be defined for each subplot, using the « **Background Color** » drop-down list (up to 16 colors may be used).

The axis color may be defined for each subplot, using the « **Axis Color** » drop-down list (up to 16 colors may be used).

The grid option for each subplot may be set/reset using the « **Grid** » check button.

You may also define the X-axis label and measurement unit, by editing the « **X-Axis** » edit control field and respectively, by selecting the measurement unit from the associated drop-down list of possible units.

Note:	The default X-axis unit is « h », representing the sampling time unit. If this value is used, all the time (abscissa) values on the curves will represent the storage index of that curve (i.e. the sampling index).
-------	---

Use the « **Default** » button to reset all the selected measurement units for the curves, and to impose the « **h** » unit for the X-axis (time axis).

Use the « **Apply** » button to effectively apply the defined settings and see their effect on the graphical window of **DSPMOT**, without closing the **Graphics | Plot Option dialog**.

Use the « **OK** » button to effectively apply the defined settings and exit back to the graphical window of **DSPMOT**, by closing the **Graphics | Plot Option dialog**.

Use the « **Cancel** » button to cancel all the defined settings and exit back to the graphical window of **DSPMOT**, by closing the **Graphics | Plot Option dialog**.

Use the « **Help** » button to get on-line information about the functions and the use of the **Graphics | Plot Option dialog**.

Hot-key : [Alt-G P]

6.2.2.3 Graphics | Arrange menu command

The **Graphics | Arrange** menu command allows you to define the position of the subplots on the graphical window of **DSPMOT**. The command is effective if more than one subplot are defined. The following options are available :

- **Auto** : use a default disposal of the subplots, depending on their number (2, 3 or 4).

- **Horizontal** : the plot window is divided in horizontal regions for sub-plotting. The subplots are displayed in a row, from left to right, on the graphic window.
- **Vertical** : the plot window is divided in vertical regions for sub-plotting. The subplots are displayed one below the other.
- **Mixed** : (applied for the **3-subplots** case): two subplots are displayed in the upper half of the plot window, and the third in the lower half.

6.2.2.4 Graphics | Zoom menu command

The **Graphics | Zoom** menu command allows you to select fixed zoom areas of the **first subplot** on the graphical window of **DSPMOT**. The following options are available :

- **In** : zoom-in the graphical image of the first subplot
- **Prev** : zoom-out one step the graphical image of the first subplot
- **Out** : zoom-out back to the initial graphical image of the first subplot

In order to freely zoom any graphical image, you may use the mouse to select a part of the current subplot, allowing the zooming of the selected region. The selection is done by **pressing the left mouse button** and **dragging** the zoom cursor on the display surface (the movement is bound to the area of the subplot). On the release of the mouse button, the selected region is expanded to the dimension of all the subplot. Successive zooms may be applied to any of the subplots.

Note that, when moving the mouse cursor, you can see, at the bottom of the graphic window, the coordinates on the left and right axes of the current cursor position on the screen. Thus, measurements may be done on the plots. If no region is selected for zooming, the plot is unchanged.

Use a double-click of the left mouse button, with the mouse in the graphical area of a subplot, in order to zoom-out one level back from the currently displayed image.

6.2.2.5 Graphics | Refresh menu command

The **Graphics | Refresh** menu command allows you to refresh all the drawings on the graphical window of **DSPMOT**. Use this command if the graphical image was not correctly redrawn and contains apparently erroneous graphical drawings.

6.2.2.6 Graphics | Export to WMF menu command

The **Graphics | Export to WMF** menu command will be used to save the actual graphic window contents on a file on the system disk, into a special standard format, the **Windows Metafile Format** (or **WMF**). A special dialogue is opened, similar to the **Save As ...** one, which asks you to indicate the name of the metafile file (its default extension is **“.WMF”**). The saved file may then be **imported** in other Windows applications which have adequate **graphic filters** and recognize the **metafile format**. Thus, the graphics may be included in other documents, more text may be added to the plots, colors and other features may be changed. See adequate documentation for more information about metafile files, and the user manuals of specific programs in order to find out their capability to import metafile files.

6.2.2.7 Graphics | Export to ASCII menu command

The **Graphics | Export to ASCII** menu command will be used to save the actual values of all the uploaded variables values, on a file on the system disk, into a standard **ASCII** text format. A special dialogue is opened, similar to the **Save As ...** one, which asks you to indicate the name of the **ASCII** file (its default extension is **“.txt”**). The saved file may then be examined, and also read and imported in different other programs as Excel, Word, etc.

6.2.3 Motion menu commands

If at least one motion project is opened in the **DSPMOT** program, the **Motion** menu command has the following menu sub-commands (see Figure 6.7):

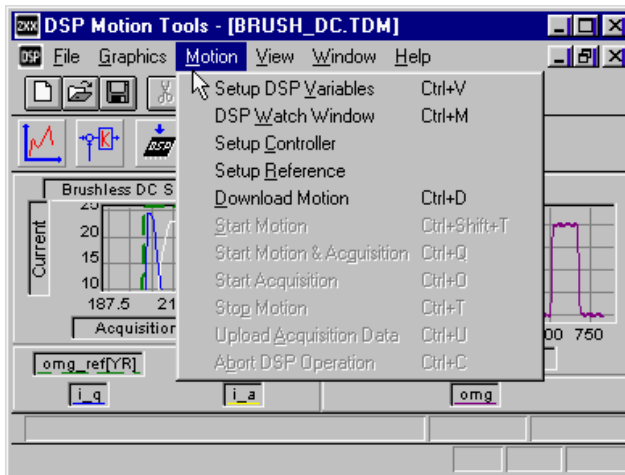


Figure 6.7. Motion menu command

- Setup DSC Variables
- DSC Watch Window
- Setup Controller
- Setup Reference
- Download Motion
- Start Motion
- Start Motion & Acquisition
- Start Acquisition
- Stop Motion
- Upload Acquisition Data
- Abort DSC Operation

Depending on the state of the motion project, some of these menu sub-commands will be enabled or not, allowing you to execute only the allowed operations for a given project environment.

Hot-key : [Alt-M]

description of each of these menu sub-commands.

The following paragraphs give the complete

6.2.3.1 Motion | Setup DSC Variables menu command

The **Motion | Setup DSC Variables** menu command allows you to attach to a given motion project a specific DSC program (its *.out and *.map files), as well as to examine, initialize and select for data logging the variables of that application.

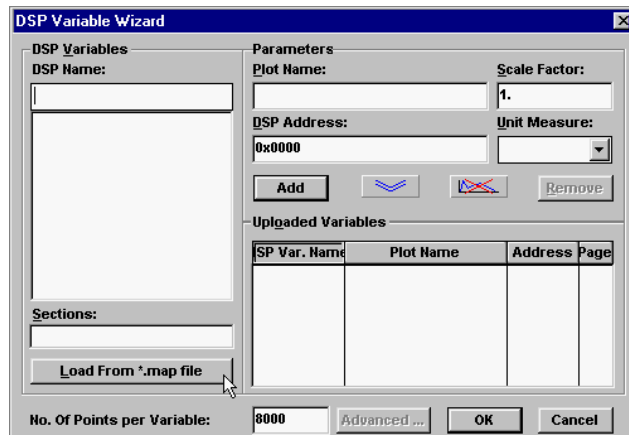


Figure 6.8. Motion Setup DSC Variables dialog, for a new motion project

For a new project, the **Motion | Setup DSC Variables** command will open the dialog presented in Figure 6.8.

In this case, the first command that you MUST execute is to attach to the new project the DSC application program files. In order to do this, you need to press the « **Load From *.map File** » button. This command will open a standard Windows « **File Open** » dialog, allowing you to browse and select the *.map file of his DSC application. Once selected, the *.map file information is loaded by **DSPMOT** and allows you to apply all the other options of the **DSPMOT** program.

Important notes:	1). You MUST generate, at the linking stage of your DSC application code generation, the associated *.map file, in order to be able to visualize the variables list and to activate all the features of the DSPMOT program. Use the corresponding linker options in order to generate the *.map file (see Texas Instruments reference manuals).
	2). The executable file of the DSC application (the *.out file), associated to the *.map file, MUST be located in the same directory as the *.map file, in order to be able to download and execute the program from the DSPMOT environment.

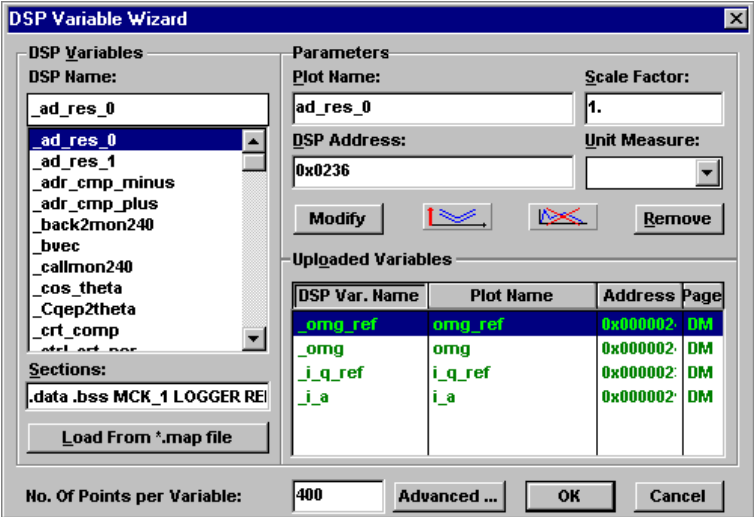


Figure 6.9. Motion Setup DSC Variables dialog, after loading a *.map file

Once a *.map file was located and loaded using the « **Load From *.map File** » button, the « **DSC Variables** » box of the **Motion Setup DSC Variables dialog** will be filled with all the global symbols defined in the DSC application. Figure 6.9 presents an example in this case.

If you want to have only specific symbols listed in the « **DSC Variables** » list, it is possible to filter the variables list. After the first loading of the *.map file, the « **Sections** » edit control will contain all the defined data sections of the DSC program. You may eliminate any of these sections from this list, and then load again the *.map file (using the same « **Load From *.map File** » command). After the re-

loading of the *.map file, only the symbols belonging to the selected sections will be displayed in the « **DSC Variables** » list.

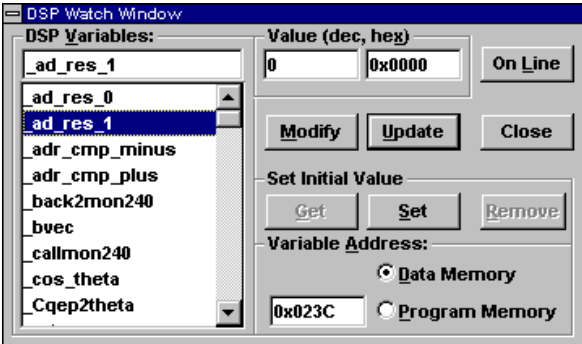


Figure 6.10. Motion DSC Watch dialog

The main purpose of the **Motion Setup DSC Variables dialog** is to define the variables which will be stored during the motion, using the data logging feature included in the DSC program. If the DSC application has included the data logger feature (see par. 6.3.2 for details on how to include the data logger features in your program), you may select in this dialog all the variables to be stored, and define their parameters needed for this purpose.

Important remarks :

1. If the data logger module is **NOT** included in the DSC application, any selection and settings done in the **Motion Setup DSC Variables dialog** will have no effect as concerns the data uploading from the DSC : no data will be stored during the motion, and correspondingly, no data can be uploaded from the **MSK28335** board and plotted by the **DSPMOT** program.
2. You may insert in the sections filter list names of the program sections. Be aware that after re-loading the symbols list from the **.MAP** file, the displayed list contains not only the variables symbols (defined in the data memory sections of the DSC memory), but also the functions and constants symbols (defined in program memory sections of the DSC memory). The attempt to store those « variables » during the motion will have as a result the plot of a constant value, representing in fact the numerical « value » of the DSC program code located at the address of that symbol.

Any of the variables from the « **DSC Variables** » list may be selected by you. Once a variable is selected (outlined), the program display its address as found in the ***.map** file, in the « **DSC Address** » edit control.


At the same time, the program sets, as default graphical name for that variable, the same name as that listed in the ***.map** file. This name is displayed in the « **Plot Name** » edit control, and may be changed by you to a more suggestive one.

The variables loaded from the DSC board represent 16 bits integer values. They are considered as signed integer numbers, without a measuring unit, for the graphical representations in **DSPMOT**. Their range is between **-32768** and **+32767**. Usually, their graphical representation will be done in these coordinates, allowing you to examine the real computational behavior of the DSC algorithms. If a physical representation of these numbers is needed, the « **Scale Factor** » edit control allows you to define a scaling factor for the selected variable. By default set to **1**, this scaling factor may be set to another positive value, which must correspond to the displayed measurement unit, chosen for the graphic representations of the variable, in the « **Unit Measure** » edit control. It is yours responsibility to determine a scale factor which is coherent in relation with the measurement unit chosen for the same variable in the « **Unit Measure** » edit control.

You may also modify the address of a selected variable, if wanting to define a new location to be watched or logged during the motion. In this case, the « **DSC Address** » edit control may be directly changed, together with other parameters (as « **Plot Name** », « **Scale Factor** » or « **Unit Measure** »).

In order to validate any change done in one or more of the « **DSC Address** », « **Plot Name** », « **Scale Factor** » or « **Unit Measure** » controls, you must push the « **Modify** » button.

In order to remove a variable from both the « **DSC Variables** » list as well as from the « **Uploaded Variables** » table, use the « **Remove** » button.

Once a variable is selected in the « **DSC Variables** » list, it may be added to the logger data list, grouped in the « **Uploaded Variables** » table. You may add it to the list using the « **Add** » button . The table lists the DSC variable name, the plot variable name, its address and DSC memory page.

In order to remove a variable only from the « **Uploaded Variables** » table, use the « **Remove from the list** » button .

Another parameter which may be defined in the **Motion Setup DSC Variables dialog** is the number of data acquisition points which will be stored during the motion. This parameter may be defined in the « **No of points per Variable** » edit control. You may change this value, according to the number of selected variables in the « **Uploaded Variables** » table and with the needed data acquisition time frame. The total memory required to store all the selected variables for the selected number of points is checked by **DSPMOT**, as compared with the specific data logger memory section which must be defined in the DSC application program. If the available memory is smaller than the required one, the program warns you and require the modification of the « **No of points per Variable** » parameter in order to fit the available memory limits.

Use the « **OK** » button to effectively apply the defined settings and exit back to the graphical window of **DSPMOT**, by closing the **Motion Setup DSC Variables dialog**.

Use the « **Cancel** » button to cancel all the defined settings and exit back to the graphical window of **DSPMOT**, by closing the **Motion Setup DSC Variables dialog**.

Use the « **Help** » button to get on-line information about the functions and the use of the **Motion Setup DSC Variables dialog**.

Hot-key : [Ctrl-V] or [Alt-M V]

6.2.3.2 Motion | DSC Watch Window menu command

The **Motion | DSC Watch Window** menu command allows you to examine, initialize and on-line modify the variables of the DSC application. Figure 6.10 presents the dialog opened when the command is executed.

The « **DSC Variables** » list contains the list of symbols which were loaded from the *.map file associated to the DSC application, in the **Motion Setup DSC Variables dialog** (see. par. 6.2.3.1 for details).

The main purpose of this dialog is to allow the examination and/or modification of the DSC program variables from the data or program memory. These operations may be done before the execution of the DSC program, in order to set the initial values of some program variables or constants. This window may also be opened after the start of the DSC motion application (using the « **Motion | Start Motion** » command), **IF** you have properly included in your DSC application the communication module supplied with the **MCK28335** kit (see par. 6.3.1 for details about how to include the communication module of **MSK28335** board into your applications). **If this communication module is active in the DSC application**, during the run of the DSC program, you will be able, from the **Motion DSC Watch dialog** to also examine and / or modify on-line, the contents of the DSC memory contents.

You may select any of the symbols from the « **DSC Variables** » list. Once a variable from the list is selected (outlined), the program will display its address and location (data or program memory), in the « **Variable Address** » group box.

If you press the « **Update** » button, the **DSPMOT** program will communicate with the DSC board and read the contents of the selected memory address. The contents of that memory location will be loaded and displayed in the « **DSC Value (dec, hex)** » group box, in both decimal as well as hexadecimal formats.

If you want to modify the value of the selected variable, you simply introduce the new value in the corresponding edit controls, in decimal or hexadecimal form, respectively, and then press the « **Modify** » button. The **DSPMOT** program then sends the new value to the DSC board and store it at the address where the variable is located.

The **Motion DSC Watch dialog** has also another important feature: it may be used in order to set the initial values of specific variables of the DSC program. Thus, you may set these initial values, and **DSPMOT** will set them accordingly, each time the motion program is started. These values are saved when the motion project is saved, and reloaded each time the project is reloaded (using the **File | Open** menu command).

The setting of initial values for the variables may be done using the « **Set Initial Value** » group box buttons. Thus, once a variable is selected in the « **DSC Variables** » list, and a value is set for it (using the « **Modify** » button), you may set this value as the initial value by pressing the « **Set** » button from the « **Set Initial Value** » group box. The variable will be outlined with a green color in the « **DSC Variables** » list. All the variables which have initial values set, are grouped in alphabetic order at the beginning of the « **DSC Variables** » list.

If you select in the « **DSC Variables** » list a variable which has an initial value set, this initial value may be obtained using the « **Get** » button.

The initial value set of a variable may be eliminated by selecting that variable in the « **DSC Variables** list and pressing the « **Remove** button from the « **Set Initial Value** group box.

Important remark : The attempt to modify in the program memory space the value of a symbol may lead, if the selected symbol is not a data but the name of a program element (function, label, etc.), to the modification of the DSC program code itself. In this case, the system behavior will be unpredictable and may lead to completely unsatisfactory results, and eventually to the complete blocking of the DSC board. In these cases, you need to reset the **MSK28335** board in order to reestablish the proper communication between the **DSPMOT** program and the DSC board and to reload the DSC program.

The « **On Line** » button will open a small dialog which will permanently display the up-dated value of the currently selected variable. During the execution of the DSC program, this option allows you to continuously evaluate the value of the variable. This command will also close the **Motion DSC Watch dialog**.

Note that the « **On Line** » opened dialog is a modal dialog, i.e. in order to accede to any of the other commands of **DSPMOT**, you need first to switch this dialog to the **Motion DSC Watch dialog**, by pressing the « **Stop** » button. Thus, only one variable may be watched at a time using this command. See par. 6.2.5.5 for details about the on-line visualization of more than one variable at the same time.

Use the « **Close** » button to close the **Motion DSC Watch dialog** and exit back to the graphical window of **DSPMOT**.

Hot-key : [Ctrl-M] or [Alt-M W]

6.2.3.3 Motion | Setup Controller menu command

The **Motion | Setup Controller** menu command allows you to define, examine and / or modify the motion controllers parameters for the DSC application.

Figure 6.11 presents the dialog opened when the command is executed.

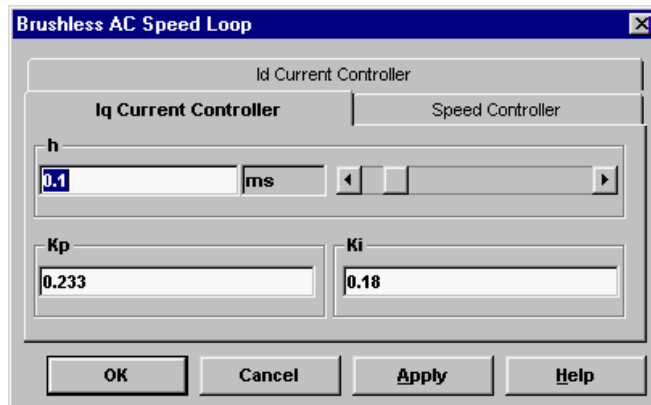


Figure 6.11. The Motion Setup Controller dialog

The main purpose of this dialog is to allow the examination and/or modification of the parameters of the digital controllers implemented on the DSC board.

These operations may be done before the execution of the DSC program, in order to set the initial values of the controllers parameters. The settings performed in this dialog will be effective only **IF** you have properly included in your DSC application the *controllers module* supplied with the **MCK28335** kit (see par. 6.3.4 for details about how to include the controllers module of **MSK28335** board into your applications).

Note that this dialog allows you to set the parameters for up to three specific controllers, i.e. one speed controller and two current controllers. All these controllers are

supposed to have a **discrete PI structure**. Specific functions names and controllers parameters are supposed to be defined in you DSC application, in order to correctly transfer the parameters settings to the DSC board. If the **DSPMOT** program does not find these variables defined in the DSC application (in the variables list from the *.map file), this command is not accessible to you.

An important advantage of this dialog is that the controllers parameters may be given as computed by you from the tuning procedures of the controllers. Thus, the **DSPMOT** program will compute, based on these values, the scaled values and the corresponding scaling factors for the DSC implementation of the discrete PI control schemes. Also the digital control sampling periods will be defined in milliseconds.

For each of the controllers, you may define the following parameters:

- **h** : the sampling period for the selected controller, expressed in milliseconds. Based on this value, **DSPMOT** will compute the required parameters in order to properly set the real time interrupts on the **MSK28335** board. As a remark, the speed sampling period may be different from the current loops sampling period. As concerns the current loop(s), the sampling period will be the same in the case of two current loops. See par. 6.3.4 for details about the DSC implementation related to the digital controllers.
- **Kp, Ki** : the proportional, respectively the integral constant of the discrete PI controller. As stated before, these values are converted by **DSPMOT**, for the DSC program level, into sets of two parameters, the scaled values (normalized in **IQ16** format), and the associated scaling factors.

Note that you may freely define your own control scheme and functions, included in the DSC program. In this case, it is possible to set the controllers parameters using the other commands of **DSPMOT** as, for example, **Motion DSC Watch Window dialog**, or **Motion Setup DSC Variables dialog**.

Use the « **Apply** » button to effectively apply the defined settings in **DSPMOT**, without closing the **Motion | Setup Controller dialog**.

Use the « **OK** » button to effectively apply the defined settings and exit back to the graphical window of **DSPMOT**, by closing the **Motion | Setup Controller dialog**.

Use the « **Cancel** » button to cancel all the defined settings and exit back to the graphical window of **DSPMOT**, by closing the **Motion | Setup Controller dialog**.

Use the « **Help** » button to get on-line information about the functions and the use of the **Motion | Setup Controller dialog**.

Hot-key : [Alt-M C]



DSC toolbar icon :

6.2.3.4 Motion | Setup Reference menu command

The **Motion | Setup Reference** menu command allows you to define, examine and / or modify the motion reference parameters for the DSC application.

Figure 6.12 presents the dialog opened when the command is executed.

The main purpose of this dialog is to allow the examination and/or modification of the values of the reference generator block implemented on the DSC board.

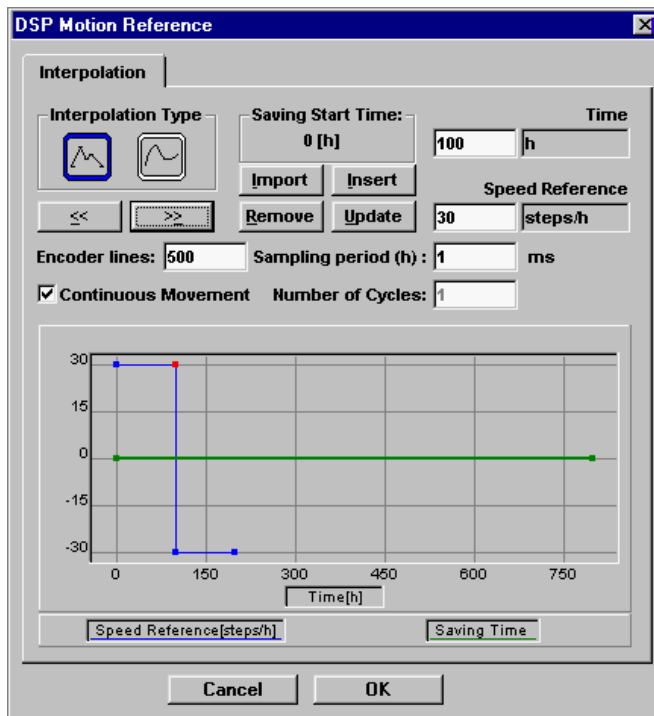


Figure 6.12. The Motion Setup Reference window

These operations may be done before the execution of the DSC program, in order to set the initial values of the reference generator parameters. The settings performed in this dialog will be effective only **IF** you have properly included in your DSC application the *reference module* supplied with the **MCK28335** kit (see par. 6.3.3 for details about how to include the reference module of **MSK28335** board into your applications).

Note that this dialog allows you to set a linearly interpolated reference generator for the DSC program. Specific functions names and reference parameters are supposed to be defined in your DSC application, in order to correctly transfer the parameters settings to the DSC board. If the **DSPMOT** program does not find a specific data memory segment defined in the DSC application (in the *.map file), this command is not accessible to you.

An important advantage of this dialog is that the reference values may be defined at an advanced level in the Windows environment. Then, the **DSPMOT** program will compute, based on these values, the scaled values for the DSC

implementation of the linear interpolation reference generator schemes and will load them on the DSC board.

The **Motion Setup Reference** dialog is used to generate a reference described by **points**, using a **linear** or **spline** interpolation method for intermediate values of time.

Note that the actual implementation of the reference generator module included on the **MCK28335** DSC library software contains only the **linear** interpolation reference.

In principle, the reference generator module is associated to a speed reference function. However, you must be aware that you may use the same function as another type of reference generator (for example as a position reference generator).

The **interpolation** type may be selected as **linear** (straight lines between the chosen points), or **spline** (cube spline, using 3rd order polynomials), in the **interpolation** control box. Note that the latter is not yet implemented on the DSC board.

The curve is described by points. For each point of the curve, the **X**-axis variable (**Time**), and **Y**-axis variable (the **Speed Reference**), are successively defined by you. An automatic display of the already defined reference curve is plotted on the screen. The graphic is auto-resizable, to the actual limits of the defined points. Using only the keyboard, you may set the values by entering the numerical values and pressing the **<ENTER>** key. At this point you will notice the red point in the gray graphical area of the dialogue. The caret is now blinking in the **Time edit box**, ready for the insertion of a new point.

You may also select a point on the curve, using the mouse in the *graphic region* of the window, by clicking on it with the left mouse button. Once a point is selected, the program enters in the **update mode**. A selected point is red colored, and its coordinates appear in the **Time** respectively **Speed Reference edit boxes**. **Update mode** means that by changing the values of a point in the corresponding *Speed Reference edit* it will update just by pressing the **<ENTER>** key. One can also *update* a point by setting its new coordinates and then clicking the **Update button**.

If you modify also the **Time** edit value, pressing the **<ENTER>** key will be treated as an « **Insert** » command if no other point has the same **Time** value, or as an « **Update** » command if there is another point having the same **Time** value.

An warning is issued if you try to insert a new point having the same **Time** value as an already defined one, by pressing the « **Insert** » button.

A selected point may also be displaced by you by dragging it with the left mouse button pressed to a new position in the graphical area of the reference window.

The *currently selected point* may be removed (deleted) using the « **Remove** » button in the window.

The **left <<** and **right >>** buttons, will change the currently selected point while in update mode.

The « **Import** » button may be used in order to load values from data files, compatible with *.TXT EXCEL files (separator method: TAB). Thus, different data sets may be loaded (even concatenated) directly from the file.

The « **Continuous Movement** » check box indicates if the defined motion cycle must be indefinitely repeated in the DSC program.

The « **Number of Cycles** » edit control allows you to specify the number of repetitions of the motion cycle before stopping the motion on the DSC board. Note that this option is disabled if the « **Continuous Movement** » check box is selected.

The time edit control has as pre-selected measurement unit the discrete sampling period h. It means that time value for each point is expressed as the number of elapsed sampling periods. On each sampling period, the reference generator function will be called.

The speed reference edit control has as pre-selected measurement unit the steps/s. It means that each unitary value on the Y-axis represents one bit of the numerical representation of the reference value, considered as a signed 16 bits integer, ranging between -32768 and +32767.

By double-clicking the associated units edit control, a combo box is displayed, allowing you to select other measurement units for the time, respectively for the reference values. WARNING : these settings do not affect the numerical representation of the reference values in the DSC program !

You may examine and modify the number of lines of the position encoder transducer, used in the **MSK28335** board. Combined with the value of the sampling period of the speed control loop (displayed also in this window, but not modifiable in this dialog), this parameters allows that the **DSPMOT** program computes the reference table for the speed controller.

Notes:	1). Up to this point, the <u>maximum accepted number of points</u> in a reference curve is 100.
	2). You may freely define your own reference generator functions, included in the DSC program. In this case, it is possible to set the reference generator parameters using the other commands of DSPMOT as, for example, Motion DSC Watch Window dialog, or Motion Setup DSC Variables dialog.

Use the « **OK** » button to effectively apply the defined settings and exit back to the graphical window of **DSPMOT**, by closing the **Motion | Setup Reference dialog**.

Use the « **Cancel** » button to cancel all the defined settings and exit back to the graphical window of **DSPMOT**, by closing the **Motion | Setup Reference dialog**.

Use the « **Help** » button to get on-line information about the functions and the use of the **Motion | Setup Reference dialog**.

Hot-key : [Alt-M R]

DSC toolbar icon :



6.2.3.5 Motion | Download Motion menu command

The **Motion | Download Motion** menu command allows you to download to the DSC board the application associated to the currently defined *.MOT file. The contents of all the initialized data and program sections of the *.OUT file is send to the DSC board. This command must be executed before starting the motion.

Notes:	1). Use this command after the opening of a *.MOT file, as well as after the start of a new *.MOT project. Also download the code each time a modification was done in the DSC code (a new *.OUT file was generated).
	2). Do NOT download again the motion program, if modifying any or more of the following elements: <ul style="list-style-type: none"> • the graphical parameters and settings • the data logging variables • the reference curve • the controllers parameters Instead, simply re-start the motion and perform data acquisition, display it, etc.

Hot-key : [Ctrl D] [Alt-M D]



DSC toolbar icon :

6.2.3.6 Motion | Start Motion menu command

The **Motion | Start Motion** menu command allows you to start the execution of a previously downloaded DSC program to the **MSK28335** board.

By using this menu command, the motion is started without performing the data acquisition, too. This allows you to use the separate **Motion | Start Acquisition** menu command, in order to start the data acquisition at a latter moment. This option may be useful for measurements performed after the initial transients of the motor, in order to trace the steady-state regime of the motor.

For the simultaneous start of the motion and of the data acquisition process, use the **Motion | Start Motion & Acquisition** menu command (par. 6.2.3.7).

Important Note:	The data acquisition, reference generator, controllers parameters setting and stop motion features as well as the DSPMOT commands may be used only if the compatible functions are included in the DSC application code. See par. 6.3 for details on how to include these features in your applications.
------------------------	---

Hot-key : [Ctrl Shift T] [Alt-M S]

6.2.3.7 Motion | Start Motion & Acquisition menu command

The **Motion | Start Motion & Acquisition** menu command allows you to start the execution of a previously downloaded DSC program to the **MSK28335** board, as well as to start the data acquisition process.

By using this menu command, the motion and the data acquisition process are started at the same time. This option is useful for measurements performed from the starting of the motion, and allows one to trace the starting transient regime of the motor.

For the separate start of the motion and of the data acquisition process, use the **Motion | Start Motion** menu command (par. 6.2.3.6) combined with the **Motion | Start Acquisition** menu command (par. 6.2.3.8).

Important Note:	The data acquisition, reference generator, controllers parameters setting and stop motion features as well as the DSPMOT commands may be used only if the compatible functions are included in the DSC application code. See par. 6.3 for details on how to include these features in your applications.
------------------------	---

Hot-key : [Ctrl Q] [Alt-M Q]

DSC toolbar icon :



6.2.3.8 Motion | Start Acquisition menu command

The **Motion | Start Acquisition** menu command allows you to start the data acquisition process on the **MSK28335** board.

This menu command may be given only after a **Motion | Start Motion** menu command was given. By using this menu command, the data acquisition process is started after the start of the motion. This option may be useful for measurements performed after the initial transients of the motor, in order to trace the steady-state regime of the motor.

For the simultaneous start of the motion and of the data acquisition process, use the **Motion | Start Motion & Acquisition** menu command (par. 6.2.3.7).

Important Note:	The data acquisition, reference generator, controllers parameters setting and stop motion features as well as the DSPMOT commands may be used only if the compatible functions are included in the DSC application code. See par. 6.3 for details on how to include these features in your applications.
------------------------	---

Hot-key : [Ctrl Q] [Alt-M Q]

6.2.3.9 Motion | Stop Motion menu command

The **Motion | Stop Motion** menu command allows you to stop a motion which executes on the DSC board. You may stop a motion cycle before completion or, for continuous movement mode (see par.6.2.3.4), it is the only way to stop a previously started motion.

This menu command may be given only after the start of the motion, using one of the **Motion | Start Motion & Acquisition** or **Motion | Start Motion** menu commands. It will stop the execution of the DSC program, **ONLY IF** a specific variable, named "**Stop**", is defined and tested by the DSC program during the execution of the motion (see par. 6.3 for details on how to include these features in your applications).

Hot-key : [Ctrl T] [Alt-M P]

DSC toolbar icon :



6.2.3.10 Motion | Upload Acquisition Data menu command

The **Motion | Upload Acquisition Data** menu command allows you to transfer the acquired data from the **MSK28335** board to the PC computer, and to display it, accordingly to the actual plot settings.

This menu command may be given only after a **Motion | Start Motion & Acquisition** or **Motion | Start Acquisition** menu command was given. By using this menu command, the acquired data may be up-loaded on the PC and visualized.

If the data acquisition process (started before this command) was not yet completed, you are prompted to decide if to stop it, or to continue the motion in order to finish the data logging process.

During the upload process, the data is gradually displayed in the graphic sub-plots, conform to the plot actual settings. An automatic re-sizing of the plot range is done by the **DSPMOT** program during this operation.

Important Note:	The data acquisition, reference generator, controllers parameters setting and stop motion features as well as the DSPMOT commands may be used only if the compatible functions are included in the DSC application code. See par. 6.3 for details on how to include these features in your applications.
------------------------	---

Hot-key : [Ctrl U] [Alt-M A]



DSC toolbar icon :

6.2.3.11 Motion | Abort DSC Operation menu command

The **Motion | Abort DSC Operation** menu command allows you to stop the up-loading process of the acquired data from the **MSK28335** board. This command may be used to stop a data transfer process if you do not need to examine the remaining not-transferred data from the DSP. Anyway, the already transferred data may be displayed and analyzed on the PC.

Important Note:	The data acquisition, reference generator, controllers parameters setting and stop motion features as well as the DSPMOT commands may be used only if the compatible functions are included in the DSC application code. See par. 6.3 for details on how to include these features in your applications.
------------------------	---

Hot-key : [Ctrl C] [Alt-M B]



DSC toolbar icon :

6.2.4 View menu commands

The **View** menu command may be used in order to select or deselect the view of the toolbar or of the keyboard status bar of the **DSPMOT** window.

The **View** menu command has the following menu sub-commands:

- **View | Toolbar**
- **View | Status Bar**

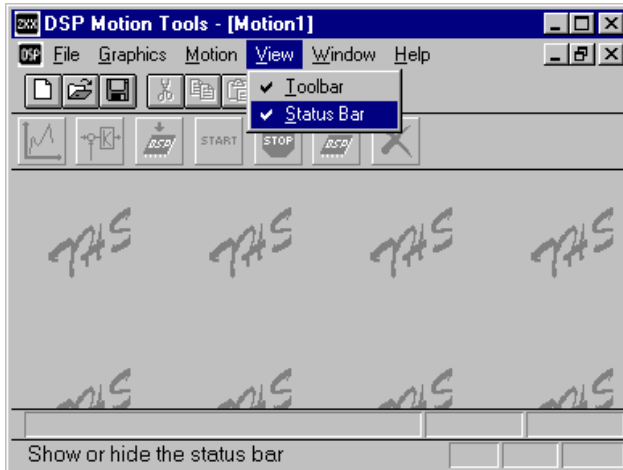


Figure 6.13 presents the **View** menu command.

Figure 6.13. The View menu command

6.2.4.1 **View | Toolbar** menu command

The **View | Toolbar** menu command allows you to show / hide the program motion toolbar. This command may be used in order to better organize the working space on the **DSPMOT** window. By default, the motion toolbar is showed on the screen and may be used to easier accede the basic motion commands of **DSPMOT**.

Hot-key : [Alt-V T]

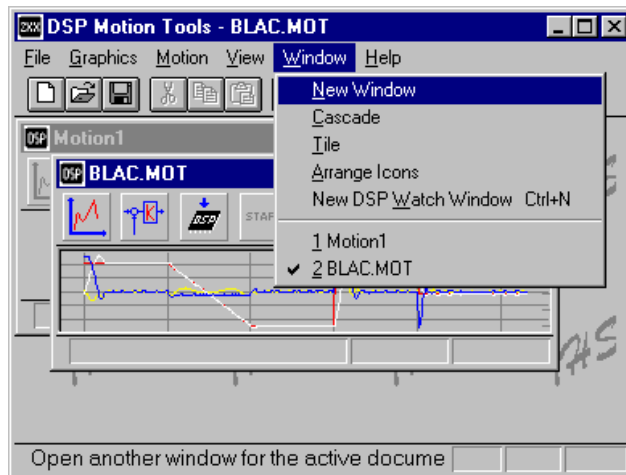
6.2.4.2 **View | Status Bar** menu command

The **View | Status Bar** menu command allows you to show / hide the PC keyboard status bar. This command may be used in order to better organize the working space on the **DSPMOT** window. By default, the PC keyboard status bar is shown on the screen.

Hot-key : [Alt-V S]

6.2.5 **Window** menu commands

Figure 6.14 presents the **Window** command menu.



The **Window** menu command allows you to rearrange the lay-out of the different motion projects opened at a moment in the program.

The **Window** menu command has the following menu sub-commands:

- **Window | New Window**
- **Window | Cascade**
- **Window | Tile**
- **Window | Arrange Icons**
- **Window | New DSC Watch Window**

It also contains the list of the up to four last opened *.MOT projects.

Figure 6.14. Window command menu

6.2.5.1 Window | New Window menu command

The **Window | New Window** menu command open another window for the active document. The new opened window is identical with the previous one. This command allows you to duplicate a motion project, in order to compare different parameters settings. On the different instances of the same project, you may test different shapes of the reference, different controller parameters, different data logging settings and even different graphical settings. The newly created instances of the project may be saved as different *.MOT files and used later.

Hot-key : [Alt-W N]

6.2.5.2 Window | Cascade menu command

The **Window | Cascade** menu command will put all the currently opened motion projects in a cascade arrangement in the **DSPMOT** environment. This command allows you to clearly outline all the projects which are opened, and to select one of them by a mouse click in the active area of the respective project. An alternative arrangement may be obtained using the **Window | Tile** menu command (par. 6.2.5.3).

Hot-key : [Alt-W C]

6.2.5.3 Window | Tile menu command

The **Window | Tile** menu command will put all the currently opened motion projects in a tile arrangement in the **DSPMOT** environment. This command allows you to clearly outline all the projects which are opened, and to select one of them by a mouse click in the active area of the respective project. An alternative arrangement may be obtained using the **Window | Cascade** menu command (par. 6.2.5.2).

Hot-key : [Alt-W T]

6.2.5.4 Window | Arrange Icons menu command

The **Window | Arrange Icons** menu command may be used to arrange the minimized icons of the opened motion projects, at the bottom of the **DSPMOT** window.

Hot-key : [Alt-W A]

6.2.5.5 Window | New DSC Watch Window menu command

The **Window | New DSC Watch Window** menu command opens a similar window as the **Motion | DSC Watch Window** menu command (see par. 6.2.3.2 for details about how to use this menu command). The only and main difference between the two approaches is related to the treatment of the “**On Line**” button command in the **DSC Watch Window**. For the **Motion | DSC Watch Window** menu command, once you select a variable and press the “**On Line**” button, the specific watch window which is opened is a modal window, i.e. you cannot select other menu commands until this watch window is closed. In this mode, only one variable may be watched at a moment.

On the contrary, if the watch window was opened from the **Window | New DSC Watch Window** dialog, at “**On Line**” button selection, the newly created watch window will not stop you to apply any of the **DSPMOT** menu commands. Thus, not only different commands may be given to the board, but also more watch windows may be opened simultaneously, allowing you to examine different variables during the motion.

Be aware that, due to the limited communication speed between the PC and the **MSK28335** board, the updating rate of the values from the watch windows will depend on the volume of data to be transferred between the two systems.

Hot-key : [Alt-W W]

6.2.6 Help menu command

The **Help** command menu may be used to get information about the **DSPMOT** main topics and usage.

The **Help** menu command has the following menu sub-commands:

- **Help | Index**
- **Help | Using Help**
- **Help | About DSC Motion**

6.2.6.1 Help | Index menu command

The **Help | Index** menu command may be used to accede the index list of **DSPMOT** help file. From there, you may get specific information about the different features of the program, as well as about the use of the different dialogs and menu commands.

Note that specific help information, related to the currently active dialog of **DSPMOT** may be obtained by pressing the “**Help**” button present in each of these dialogs.

Hot-key : [Alt-H I]

6.2.6.2 Help | Using Help menu command

The **Help | Using Help** menu command may be used to get general information about the use of **DSPMOT** help file.

Hot-key : [Alt-H U]

6.2.6.3 Help | About DSC Motion menu command

The **Help | About DSC Motion** menu command may be used to open the “**About**” dialog, showing the **DSPMOT** general information, including the program version number, registration information (user, serial number), and copyright information.

Hot-key : [Alt-H A]

6.3 Advanced DSPMOT features

The **DSPMOT** program is intended to offer you an advanced tool for motion control applications development and analysis. As already stated in the previous paragraphs, you may generate a DSC application using the development tools from Texas Instruments (assembler, C compiler, linker), and download it to the **MSK28335** board. Then, using **DSPMOT** commands as **Motion | DSC Watch Window**, you may evaluate and eventually modify the contents of program variables, set their initial values, etc. Finally, the DSC program may be started using the **Motion | Start Motion** menu command.

These features of **DSPMOT** may be used for any application written by you, if the application complies with the **MSK28335** board resources (uses the available board data and program memory).

Normally, if such a DSC application is started, the communication between the PC and the **MSK28335 DSC board** is interrupted. Then, the **DSPMOT** program is useless for the motion application analysis. No variable watch, data logging, reference or controller settings are possible from **DSPMOT** in such a case, during the motion.

If you want to have full access from your motion application, to all the advanced analysis tools offered by **DSPMOT**, you must include some specific pre-defined **DSPMOT**-compatible modules in your application. These modules are supplied with the **MCK28335 kits** software packages, as object files, and usually only functions calls are needed in order to activate them in your DSC program.

The next paragraphs give the needed details about what modules are needed, and how to include them in your applications, in order to be able to:

- communicate with the PC computer
- perform data logging and up-loading
- set the reference at the **DSPMOT** graphical level

Each paragraph will indicate the operating principles of the presented functions, their names, the associated input / output variables (if any), the corresponding include files for C programming, as well as the object files containing their DSC code and specific implementation requirements (as reserved data sections names and dimensions - if needed, etc.).

If considering assembler language programming, remember that all the functions which will be presented are C-callable. Please refer to Texas Instruments manuals for detailed information about how to integrate C-callable functions in assembly-written programs.

6.3.1 Communication Module

The communication between the PC computer and the **MSK28335 DSC board** is done through the RS-232 serial connection. Before starting any user application on the DSC board, this communication is possible due to the execution on the DSC board of the **MON28335** program, and of the corresponding communication functions on the PC (**DSPMOT** uses the **MONWIN32.DLL** communication functions for this purpose). Once the DSC program is launched, this communication channel is normally interrupted, **IF** you do not include in your application a communication module.

Such a communication module, compatible with **DSPMOT** program, is included in the **MCK28335 kits** software packages. Practically, all you need to do is to call from your application the monitor command

interpreter function, resident in the flash memory of the **MSK28335 DSC board**. From that moment, the communication between the PC and the **MSK28335 DSC board** may be performed and, transparently for your application, **DSPMOT** can examine and/or modify the contents of specific data memory cells.

Based on this principle, the **DSPMOT** program will be able, after the start of the DSC program, to:

- write into the data memory locations corresponding to selected variables - modification of variables as controllers coefficients, set the “**Stop**” variable (see below), etc.
- read from the data memory locations corresponding to selected variables - watch operations

The important aspect here is that the DSC program must call the monitor function periodically. Usually, the real-time applications will implement the real-time control functions (as position/speed and current controllers) in interrupt service routines, activated by one or more timers of the system. Because these routines are critical from the efficiency point of view, they are usually written in assembler language, for maximum optimization from the point of view of program code dimensions and execution speed.

On the other side, the communication functions are not so critical, and usually they may be implemented in lower-priority program loops. Usually, the program has a main loop, where specific non time-critical operations may be performed. It is then recommended to include the command interpreter monitor function call at this level of the DSC program.

Note that in order to ensure a continuous communication between the PC and the **MSK28335 DSC board**, it is not sufficient to call the monitor only once, in the DSC program. The call of the monitor function must be performed cyclically, from the not-time-critical sections of the DSC application.

6.3.1.1 Implementation aspects

C function prototypes:

-
- **cmd_interpreter:** /* **MON28335** command interpreter function */
 /* to be called from the main program loop */
 /* input arguments: - */
 /* output arguments: 0 if receives a “user data” command, else 1 */

```
#define cmd_interpreter_addr 0x00330000  
int (*callmon28335)(void)=(int(*)())cmd_interpreter_addr;
```

Reserved variables: (to not be defined or used by you in your application): no restrictions

Include files (containing functions prototypes): to be included in your DSC source code files, for compilation:

 \CDemos\Monitor\ExtCodeFlash\mon28335.h

Code Files: (to be included in your project): none

Restrictions:

- Please refer to the specific memory restrictions related to the **MONPC28X** program (see paragraph 4.4.8 for details).
- The memory sections which are needed for the command interpreter monitor program must not be used by you for your application. Failure to respect this restriction will have as effect the improper operation of the DSC application.
- Also consider that the monitor program uses the serial receive (**RXINT**) and transmit (**TXINT**) character interrupts of the SCI interface of the TMS320F28335 chip. Attempts to use them in your application will definitively compromise the communication process between the PC and the **MSK28335** board.

Processor overhead:

All the data communication protocol is interrupt-based, thus significantly reducing the overhead of the processor. All the data communication management is based on the reception buffer, handled by the specific interrupt service routine, and the transmission routine. The table below indicates the overhead introduced by these routines.

The monitor command interpreter function will handle characters from/to these buffers. In order to not block the processor, the monitor will not wait to receive all the characters for a specific monitor command. Only when all these characters were received, the monitor will effectively decode and execute the command, thus significantly reducing the processor overhead. Also the monitor command interpreter must not be called from an interrupt routine. Usually, if called from the main loop of the program, the command interpreter function will be interruptible by any of the real-time control routines.

At the same time, as stated before, all the real-time control functions must be interrupt-driven by the system timers. When generated, they will interrupt the lower priority monitor function. At the end of these interrupts, the monitor execution will be resumed.

The next table summarizes the processor overhead for the basic monitor interrupt operations.

Function	Processor cycles
Character reception (interrupt)	87 typical / 114 max.
Character transmission (interrupt)	54

6.3.1.2 Examples

- see par. 4.4.5 for an example on how to include the monitor call in a C code program
- see par. 4.4.6 for an example on how to include the monitor call in an assembler code program

6.3.1.3 Stopping a DSC program

A special convention is used by **DSPMOT** in order to properly stop the execution of a DSC program from the PC Windows environment, without the need to reset the DSC board.

Thus, **IF** the communication module is included and activated in the DSC application, and **IF** an integer variable named **stop** is defined by you in your application, then:

- at the start of the DSC application, the DSC program must initialize the **stop** variable to **0**
- when you press the “**Stop Motion**” motion toolbar button, or you give the **Motion | Stop Motion** menu command, **DSPMOT** will set, **IF DEFINED**, on the DSC board, the variable **stop**, to **1**

Thus, the DSC program may test (eventually in the main program loop, where also the monitor is called) the value of the **stop** variable. If the variable is found set to **1**, the program may be stopped, and the control given back to the **MON28335** program of the DSC board.

When the DSC program is started, **MON28335** gives the control to the DSC application. In order to properly exit from the DSC application and safely return to the **MON28335** program, you need to include, as the exit code from your program, a specific exit code sequence, described in the example below.

A possible main loop of a C program which calls the monitor and exits back to the **MON28335** program at the setting of the **stop** variable, is presented here:

```
/* Define the program exit sequence code */
```

```

int stop, ret_val_mon;

/* Main function */
void main(void)
{
    stop = 0;
    /* perform program initializations */
    /* ... */

    /* main loop */
    while (!stop)
    {
        ret_val_mon = (*callmon28335)(); /* call monitor in the main loop */

        /* perform other operations */
    }

    // Generate system reset
    DINT;
    EALLOW;
    SysCtrlRegs.WDCR = 0x000F; // Enable Watchdog and
                                // write incorrect WD Check Bits
                                // (001 in lieu of 101) to force a system reset

    asm(" NOP" );
    asm(" NOP" );
    asm(" NOP" );
}

```

Notes:	1. When including the data logging feature in the DSC program (see par. 6.3.2), you MUST use the END_APPL program sequence in order to exit the DSC application. Only after the stop of the motion and the safe return in MON28335 , the DSPMOT program will be able to correctly transfer the logged data from the DSC on the PC, and display its graphics.
	2. If the stop variable is defined in assembler instead of C language, the name of the variable must be, in assembler, <code>_stop</code> (note the leading underscore, which will assure the compatibility with C notation conventions).
	3. If the main program loop is written in assembler language, replace the END_APPL sequence presented in the previous example with the corresponding assembler instructions.

6.3.2 Built-in Data Logger Module

The built-in data logger module represents a powerful tool for motion applications analysis and debugging. Due to the fact that the motion control applications are real-time applications, the use of standard debugging techniques, as step-by-step program execution, is not very useful for these cases. The need to implement the digital control at the precise sampling frequencies imposes that the system runs at the full designed speed, and non-intrusive debugging techniques must be used instead.

A very attractive debugging alternative is to store, during the execution of the program, some of the variables, measured data, and even intermediate results of the computation, and to examine them, after the

completion of the motion. Storing selected variables is not so time consuming, and may be usually inserted in the program without significantly increasing the overhead of the processor.

Based on these considerations, the **DSPMOT** program allows one which implements these data logger features in his DSC application, to up-load data and to analyze them in the powerful graphical environment of the Windows program.

All you need to do is to include in your DSC application calls to specific data logger module (functions), included in the **MCWIN28335** and **MCWIN28335 Professional** software packages (see below). Then, the **DSPMOT Motion | Setup DSC Variables** menu command must be used to select which variables of the DSC program to be stored during the motion execution. During the motion execution, the data logging functions from the DSC board will store the data. After the motion execution, the **Motion | Upload Acquisition Data** menu command will transfer the stored data to the PC and will display it in the **DSPMOT** environment.

Note that the data logger feature may be used **ONLY** if you include in the DSC application the communication module (see par. 6.3.1 for details).

6.3.2.1 Implementation aspects

C function prototypes:

- **init_logger:** /* initializes the logger module */
/* to be called from the initialization part of the main program */
/* input arguments: - */
/* output arguments: - */

```
extern void init_logger();
```

- **logger:** /* performs the logged variables data storage */
/* to be called from the program point where you need to store data (usually in a real-time interrupt control routine) */
/* input arguments: - */
/* output arguments: - */

```
extern void logger();
```

Reserved variables: (to not be defined or used by you in your application):

```
extern unsigned long log_idx;  
extern unsigned long log_count;  
extern unsigned long log_size;  
extern unsigned long log_time;  
extern int *log_p[17];  
extern int log_table[];
```


C Include files (containing functions prototypes): to be included in your DSC source code files, for compilation:

\CDemos\Log_ref\Log_ref.h

Code Files: (to be included in your project):

\CDemos\Log_ref\Log_ref.c

Restrictions:

- The logger function will store the selected variables (from **DSPMOT** program, using the **Motion | Setup DSC Variables** menu command), in a specific data memory user section, named **LOGGER**. This section **MUST** be defined by you in your **.CMD** link command file, in order to correctly include the logger module in your application.
- The **LOGGER** section size may be defined by you by modifying the **LOG** memory block length in **PAGE 1** of DSC memory (see for example the contents of the **blac.cmd** or **bldc.cmd** files from the **demos\blac** or **demos\bldc** sub-directory of the **MCWIN28335** main directory). Be aware that for a size of N_{log} words of the **LOG** block, the maximum number of memory words were data may be stored is $(N_{log} - 13)$. **DSPMOT** will compute, based on the size of this block, the maximum number of storage points, depending on the number of variables selected to be stored. The program will impose this limit as the maximum number of points to be stored.
- The data memory region where the **LOG** block is located may be freely defined by you, according to your data memory requirements and the available data memory on the **MSK28335** board.

Processor overhead:

The logger function does not use any interrupt. The next table summarizes the processor overhead for the data logger operations.

Function	Processor cycles
Logger	$55 + 35 * (nr.variables_to_store)$

6.3.2.2 Example

The following example adds to the basic communication implementation example presented in par. 6.3.1, the data logger initialization function in the main function initialization part. It also add the call to the logger function in an interrupt control function (for example, the speed control one).

```
interrupt void speed_control()
{
    /* perform control operations */
    /* ... */

    logger(); /* store data */
}

void main(void)
{
    stop = 0;
    init_logger(); /* initialize the data logger module */

    while (!stop)
    {
        ret_val_mon = (*callmon28335)();
    }
}
```

```

// Generate system reset
DINT;
EALLOW;
SysCtrlRegs.WDCR = 0x000F; // Enable Watchdog and
                             // write incorrect WD Check Bits
                             // (001 in lieu of 101) to force a system reset

asm(" NOP" );
asm(" NOP" );
asm(" NOP" );
}

```

6.3.3 Built-in Reference Generator

The built-in reference generator module represents another powerful tool for motion applications easy implementation and testing. This module allows you to apply a linear interpolation-type reference generator for the motion system.

If this module is included in the DSC application, the **DSPMOT** program allows you to define different shapes of the reference signal at the Windows level, using the powerful reference generator built at the PC level.

All you need to do is to include in your DSC application calls to specific reference generator module (functions), included in the **MCWIN28335** and **MCWIN28335 Professional** software packages. Then, the reference may be defined at the PC level, using the graphical reference generator block of **DSPMOT**. At motion start, the **DSPMOT** program will download the corresponding parameters of the reference to the DSC board. These parameters will be used during the motion to generate the references values at the call of the DSC reference generator function.

Remark:

The reference generator feature may be used **ONLY** if you include in the DSC application the communication module (see par. 6.3.1 for details).

6.3.3.1 Implementation aspects

C function prototypes:

- **init_reference:** /* initialize the reference generator module */
 /* to be called from the initialization part of the main program */
 /* input arguments: - */
 /* output arguments: - */

```
extern void init_reference();
```

- **reference:** /* computes the reference corresponding to the actual sampling index */
 /* to be called from the program point where you need to compute the
 reference signal (usually in a real-time interrupt control routine) */
 /* input arguments: - */
 /* output arguments: returns the value of the reference at the calling sampling
 index */

```
extern int reference();
```

Reserved variables: (to not be defined or used by you in your application):

```
long ref_long;
unsigned int ref_indx;
unsigned int ref_time;
unsigned int ref_cycles;
unsigned int ref_time_val[100];
long ref_HL[100];
```

C include files (containing functions prototypes): to be included in your DSC source code files, for compilation:

```
\CDemos\Log_ref\Log_ref.h
```

Code Files: (to be included in your project):

```
\CDemos\Log_ref\Log_ref.c
```

Restrictions:

- The reference generator function will store the reference parameters (defined in the **DSPMOT** program, using the **Motion | Setup Reference** menu command), in a specific data memory user section, named **REFER**. This section **MUST** be defined by you in your **.CMD** link command file, in order to correctly include the reference generator module in your application.
- The **REFER** section size must be defined by you by setting the **REF** memory block length in **PAGE 1** of DSC memory (see for example the contents of the **blac.cmd** or **bldc.cmd** files from the **demos\blac** or **demos\bldc** sub-directories of the **MCWIN28335** main directory). The size of **REF** memory block, needed in order to be compatible with **DSPMOT** settings, is **305** words. This value will allow to store up to **100** points on the reference curve (maximum value accepted by **DSPMOT**).
- The data memory region where the **REF** block is located may be freely defined by you, according to your data memory requirements and the available data memory on the **MSK28335** board.

Processor overhead:

The reference generator function does not use any interrupt. The next table summarizes the processor overhead for the reference generator operations.

Function	Processor cycles
Reference	240

Note that the **DSPMOT** reference generator function is defined as a speed reference function (see par. 6.2.3.4 for details on the use of the reference generator dialog of **DSPMOT**). However, you may consider the reference as being any other needed reference signal: position, current, etc. In fact, as described in par. 6.2.3.4, the values which are considered at the DSC level may be signed numbers in **IQ16** (see **IQmath library** documentation from Texas Instruments) format or, alternatively, signed integer numbers ranging from **-32768** to **32767**. Depending on the program context, you may call the reference generator function and use its output as needed as a current, speed, position or any other reference signal type. This feature is very useful at the first stages of application development, when the reference generator may be used to debug inner control loops, as current loops, before implementing the outer control loops, as the speed and/or position ones. Clearly, you will need to change the reference curve values, depending on the significance of the reference type, according to the motion system characteristics and parameters (values ranges, time restrictions, etc.).

Be also careful that the time axis values of the reference curve are measured in sampling instants instead of real time values. This means that the reference generator will give the same output value after the same number of reference function calls, independently of the sampling rate at which the function is called.

6.3.3.2 Example

The following example adds to the previously presented example from par. 6.3.2, the reference initialization function call in the main function initialization part. It also add the call to the reference function in an interrupt control function (for example, in the speed control one).

```
interrupt void speed_control()
{
    /* perform control operations */
    /* ... */
    omg_ref = reference();
    /* perform control operations */
    /* ... */

    logger(); /* store data */
}

void main(void)
{
    stop = 0;
    init_logger(); /* initialize the data logger module */
    init_reference(); /* initialize the reference generator module */

    while (!stop)
    {
        ret_val_mon = (*callmon28335)();
    }

    // Generate system reset
    DINT;
    EALLOW;
    SysCtrlRegs.WDCR = 0x000F; // Enable Watchdog and
                                // write incorrect WD Check Bits
                                // (001 in lieu of 101) to force a system reset

    asm(" NOP");
    asm(" NOP");
    asm(" NOP");
}
```

7 Brushless AC (BLAC), Brushless DC (BLDC) and Induction Motor Vector Control (IMVC) Motion Demos (MCK28335 kits)

This chapter describes the structure and functions of two demonstrative brushless motion applications which are included in the **MCWIN28335 / MCWIN28335 Professional** platform. These applications allow evaluating a digital control structure for a brushless motor, driven in either DC or AC mode.

Contents

- 7.1. Basic principles of MCK28335 brushless motion demos
- 7.2. Brushless AC motion demo application
- 7.3. Brushless DC motion demo application
- 7.4. Induction Motor Vector control motion demo application

For a better understand of the basic DPS implementation aspects, we recommend our professional DSP development tools, series: S-(BL), MS-(BL), S-(IM) and MS-(IM) , which include source code / object code application examples and libraries. All packages are self-contained, so they allow you to start the evaluation and development immediately.

7.1 Basic principles of MCK28335 brushless motion demos

The **MCK28335 kits** provide a complete hardware platform for DSC motion control applications evaluation. The **MSK28335** DSC board connected with the **PM50** power module, together with the accompanying brushless motor may be used in order to implement complete motion control structures. Thus, it is possible not only to evaluate the DSC controller chip, but also to design and test different motion control algorithms.

The **MCWIN28335** and **MCWIN28335 Professional** control panel platforms contain two ready-to-run motion control applications, for the brushless motor included in the kits. These applications use the **DSPMOT** program presented in the previous chapter. Thus, all the advanced features of **DSPMOT** are integrated in these examples: Windows-level reference generator, controllers setting, graphical evaluation of the program variables, etc.

Built around a common real-time motion control kernel, the two applications implement the digital speed control of the motor. For the AC motor control mode, a vector control method is used, based on the position information given by the incremental encoder. For the DC motor control mode, the information from the Hall sensors included in the motor package is used to perform the block commutation of inverter phases.

The **BLAC** and **BLDC** motion applications may be used at two different levels of expertise of the user:

- at the beginner level, these applications may be used in order to get the first contact with a digital motion control application. The user may evaluate the effect of modifying the controller parameters (as control coefficients, sampling time), as well as different speed reference shapes. The possibility to watch the variables and to log the data during the motion allows one to analyze quite deeply the basic phenomena in the motion structure.
- at the advanced level, the user may replace in any of these applications the controllers and/or the reference generator functions with his own algorithms. Thus, based to an already defined motion kernel (all the real-time interrupt structure is configured), the user may concentrate his effort in the direction of finding and validating new control algorithms, including sensorless, adaptive or optimal control, etc.

Both **BLAC** and **BLDC** applications have some common features:

- the motor current(s) are measured as the voltage drop on the shunts placed in the low-side legs of the inverter. Synchronized with the PWM generator signals, these measured voltages give the value of the motor phases currents and may be used in order to close the discrete current loops of the motion control structure (see Appendix D3 for details about the current measurement scheme)
- the encoder information is used in order to estimate the motor speed. In order to not complicate too much the program, only a simple difference scheme is used. Thus, the speed estimate is given by the motor position variation between two sampling instances, in the discrete speed control loop
- two control loops are considered: a faster current control loop (by default at 100 μ s sampling rate), and a slower speed control loop (by default at 1 ms sampling rate)
- all the mathematical computations are done based on the **IQ16** fractional representation (for more details about **IQ16** format see **IQmath library** documentation from Texas Instruments). All the numbers are considered to be in **IQ16** format, except specific intermediate results which are in **Q31** format. Also the reference generator values use the long (32 bits) representations
- specific overflow treatment is applied to all the mathematical calculations. The control procedures are also adapted for overflow situations special treatment
- all the implemented control loops use discrete proportional - integral (**PI**) controllers. These controllers (see chapter 6 for details) use scaled control coefficients in order to cover a wide range of control parameters values

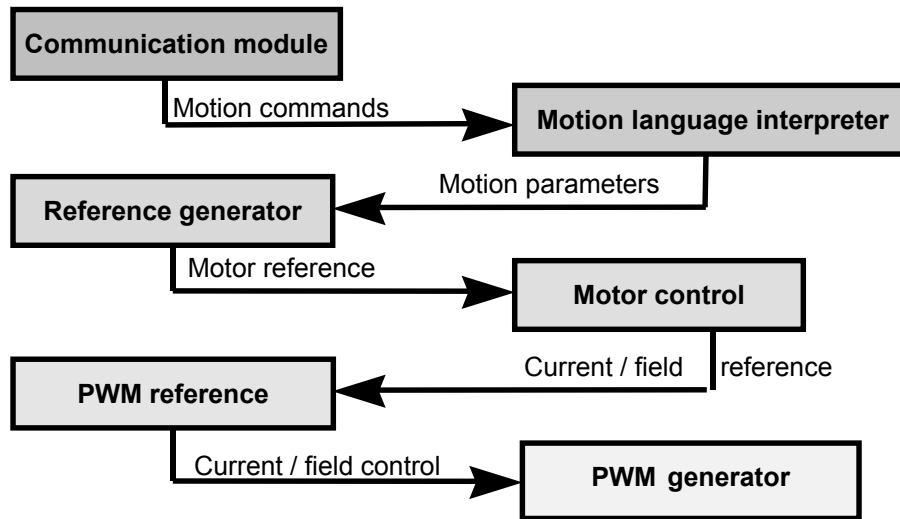


Figure 7.1. Modular software structure of the motion control applications

The two applications are based on the functional software structure presented in Figure 7.1. This structure represents in practice the basic modular approach for a standard motion control configuration.

At the top of the functional block structure, the communication module will transmit to the motion system the motion commands, generated from another computer (PC or, in the industrial applications, another control module, connected through a field bus to the DSC module). This communication module is represented, at **BLAC / BLDC** applications level, by the embedded monitor command interpreter of the **MSK28335 DSC board**.

A specific motion command interpreter, adapted to the various commands needed to drive an industrial drive will in the general case, interpret the motion commands. Based on the motion commands, the motion language interpreter will generate the motion commands towards the reference generator block. At **BLAC / BLDC** applications level, this module was not needed and is not implemented.

The reference generator block will impose the motor reference value, in **BLAC / BLDC** case the speed reference signal.

The motor reference is the input in the motor control block, containing the speed controller in the case of the **BLAC / BLDC** applications. This block will generate the reference of the quadrature current (i_q), which provides the motor torque.

The current / field control block implements the vector control (in the case of the **BLAC** application) and, respectively, the current control (in the case of the **BLDC** application). The outputs of this control block are the PWM reference signals.

Finally, the PWM reference signals are used in the PWM generator block, to drive the power inverter. A symmetric PWM generation technique is used for the two applications.

From the point of view of program structure, the **BLAC** and **BLDC** applications are configured as presented in Figure 7.2.

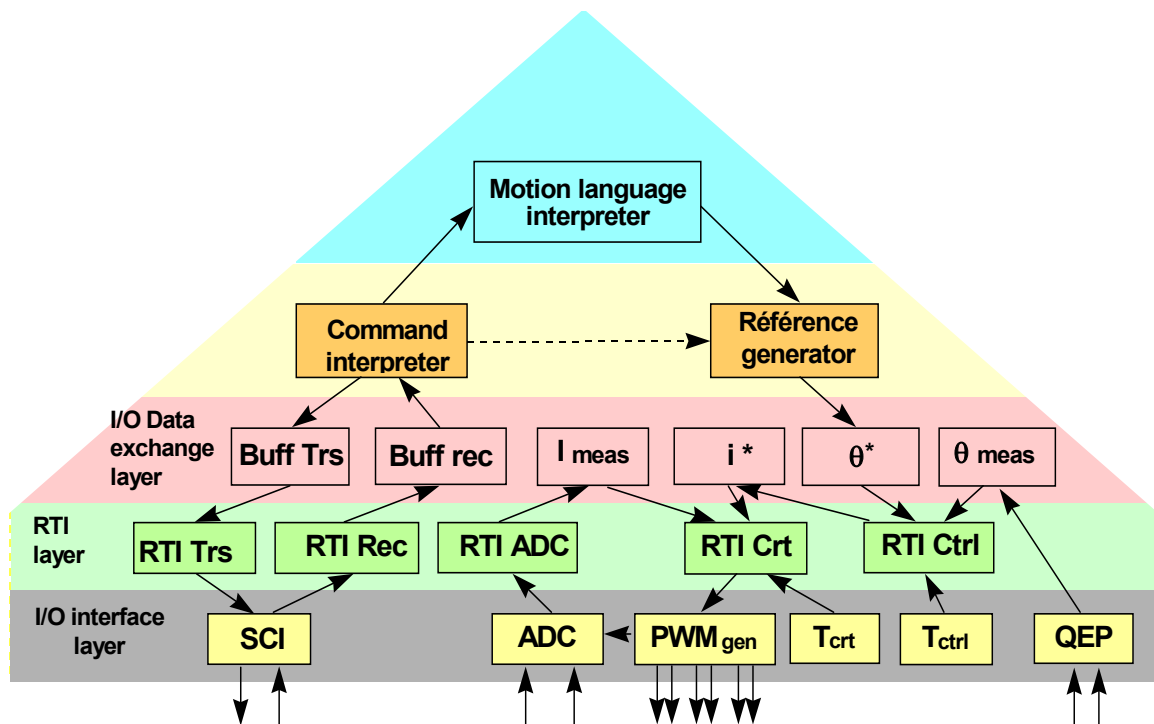


Figure 7.2. Basic real time motion kernel structure

As one can see, several different logical layers may be outlined in this configuration, which represents more or less a basic motion kernel configuration.

At the **I/O interface layer**, these applications use the following I/O resources of the TMS320F28335 chip:

- the SCI interface for the serial communication
- the ADC interface for motor phase currents measurement
- the PWM generator block for the inverter command (using Timer 1 of the DSC chip)
- the general purpose Timer 1 for current and speed control sampling periods generation (T_{crt} and T_{ctrl} respectively)
- the QEP quadrature encoder interface for position measurement from the encoder (using Timer 2 of the DSC chip)
- three input bits from the GPIO interface for Hall sensors reading

As seen from Figure 7.2, the PWM generator block will start the A/D conversion, based to the specific current measurement scheme which was implemented (see Appendix D3 for details on current measurement scheme).

At the **RTI layer level**, several real time interrupt functions are implemented:

- **RTI trs**: the serial communication transmission module. Activated by the TxINT interrupt of SCIA.
- **RTI rec**: the serial communication reception module. Activated by the RxINT interrupt of SCIA.

- **RTI ADC**: the A/D conversion measurement module. Activated by the end of conversion of the A/D converter module.
- **RTI crt**: the current control interrupt routine. Activated by the compare interrupt of Timer 1.
- **RTI ctrl**: the speed control interrupt routine. Activated by the period interrupt of Timer 1.

At the I/O data exchange layer level, several basic variables are needed:

- **Buff trs**: transmission buffer for data to be transmitted on the communication channel
- **Buff rec**: reception buffer for data received on the communication channel
- I_{meas} : measured motor current(s)
- I^* : reference motor current(s)
- θ_{meas} : measured motor position
- θ^* : reference motor position

Figure 7.2 indicates the basic logical connections between the different components of the different layers of this motion kernel.

The only differences between the **BLAC** and the **BLDC** applications are located at the level of the:

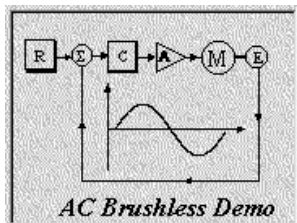
- current controller block
- PWM generator block

7.2 Brushless AC motion demo application

The brushless AC motion demo application implements a vector control method to drive the motor included in the **MCK28335 kits**.

The demo is supplied as a TMS320F28335 application, included in a *.MOT project of the **DSPMOT** program. The following files are included in the demo structure:

- **blac.out**: the COFF format executable application file
- **blac.map**: the MAP file of the application
- **blac.mot**: the **DSPMOT** project associated to the application



A mouse click on the « **Brushless AC demo** » button from the **MCWIN28335 / MCWIN28335 Professional** platform will activate the **BLAC** motion demo (the **blac.mot** project file). Executed in the **DSPMOT** program environment, this example implements a speed control loop for a three phase brushless motor, driven in AC mode. The user may define the **PI** discrete current (i_d and i_q) and speed controllers parameters, as well as the speed reference shape. At an advanced level, the user may include his own controllers and/or reference functions in the **BLAC** application and replace the demo's ones.

7.2.1 Basic structure of the control scheme for the BLAC application

The **BLAC** application control scheme is presented in Figure 7.3. As one can see, the scheme is based on the measure of two phase currents and of the motor position. The speed estimator block is, as previously stated, a simple difference block. The measured phase currents, i_a and i_b , are transformed into the stator

reference frame components, i_{ds} and i_{qs} . Then, based on the position information, these components are transformed into the rotor frame direct and quadrature components, i_{de} and i_{qe} . The speed and current controllers are, as stated before, **PI** discrete controllers. The voltage controlled oscillator block (VCO) implements (by software) the coordinates transformation and computation of the phase voltages references, v_{as}^* , v_{bs}^* and v_{cs}^* , applied to the inverter. Practically, the 6 full compare PWM outputs of the DSC controller are directly driven by the program, based on these reference voltages.

The direct current component reference i_{de}^* is set to **0**, case corresponding to the motion of the motor in the normal speed range, without considering a possible field weakening operation.

The speed reference signal is obtained using the reference generator of **DSPMOT**, included in the application (see par. 6.3.3 for details about this module of **DSPMOT**). Thus, the speed reference may be imposed at Windows level, from **DSPMOT**.

Also the controller's parameters are set in **DSPMOT**, at Windows level, from the **Motion | Setup Controller** menu command. The proportional and integral control factors, as well as the sampling periods for the current and speed control loops, are set using this **DSPMOT** command (see par. 6.3.4 for details about the control module and functions).

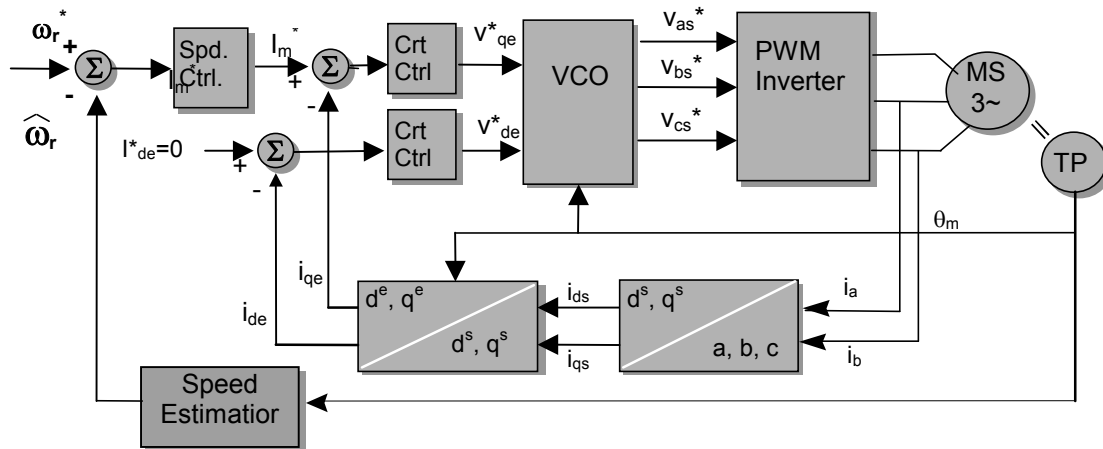


Figure 7.3. BLAC control scheme

7.2.2 BLAC demo parameters and variables

The **BLAC** application has the following control parameters, tuned for the motor included in the **MCK28335** kit:

current controllers:

	Pittman motor	Escap motor	Technosoft MBE.300E500 motor
Proportional factor - K_P	0.6	1.2	0
Integral factor - K_I	0.5	1.2	0.9525
Sampling period - h_{crt}	0.1 ms	0.1 ms	0.1 ms

speed controller:

	Pittman motor	Escap motor	Technosoft MBE.300E500 motor
Proportional factor - K_P	50	400	155
Integral factor - K_I	2.5	60	19.4
Sampling period - h_{crt}	1 ms	1 ms	1 ms

Please note that these parameters are given at **DSPMOT** program level. **DSPMOT** will automatically scale the control coefficients values in order to convert them in the **IQ16** fractional representation (for more details about **IQ16** format see **IQmath library** documentation from Texas Instruments). The sampling periods values are used to compute the compare and period values for Timer 1 of the DSC controller, if supplied with a 30MHz clock signal.

During the tests which may be performed with the **BLAC** application, the user may select, at the **DSPMOT** level, different variables of the program, in order to store them during the motion, up-load and visualize them using the graphical features of **DSPMOT**. The file **BLAC.MAP** contains the complete list of these variables, which may be selected using the **Motion | Select DSC Variables** menu command of **DSPMOT**.

We present here the significance of the most important parameters and variables which may be considered by the user for motion evaluation purposes:

Motion parameters:

Name	Type	Description
IdController.Ki	IQ16	Integral coefficient for id current controller (scaled)
IqController.Ki	IQ16	Integral coefficient for iq current controller (scaled)
SpeedController.Ki	IQ16	Integral coefficient for speed controller (scaled)
IdController.Kp	IQ16	Proportional coefficient for id current controller (scaled)
IqController.Kp	IQ16	Proportional coefficient for iq current controller (scaled)
SpeedController.Kp	IQ16	Proportional coefficient for speed controller (scaled)
_ctrl_crt_per	unsigned int	Current control period value

_ctrl_ps_per	unsigned int	Speed control period value
--------------	--------------	----------------------------

Motion variables:

Name	Type	Description
ThetaEstimator.ElectricPosition	int	electric position angle in encoder pulses, scaled for an electric period
ThetaEstimator.Theta	IQ16	electric position angle in Q15 format
Encoder.Position	int	motor position
Adc.Ia	IQ16	phase a current
Adc.Ib	IQ16	phase b current
Tabc2dq.Id	IQ16	rotor frame direct current component
Tabc2dq.IdRef	IQ16	rotor frame direct current component reference
Tabc2dq.IqRef	IQ16	rotor frame quadrature current component reference
Tabc2dq.Iq	IQ16	rotor frame quadrature current component
Adc.OffsetIa	long	offset measured on the A/D channel 6 (used for phase a current measurement)
Adc.OffsetIb	long	offset measured on the A/D channel 5 (used for phase b current measurement)
SpeedController.Ref	IQ16	motor speed reference
SpeedEstimator.Speed	int	motor speed (estimated from position variation)
pwm.UaRef	IQ16	phase a reference voltage
pwm.UbRef	IQ16	phase b reference voltage
pwm.UcRef	IQ16	phase c reference voltage
IdController.Out	IQ16	rotor frame direct voltage reference
IqController.Out	IQ16	rotor frame quadrature voltage reference

7.2.3 A BLAC example

The **BLAC** application has a predefined motion profile and parameters, which are automatically loaded at the start of the demo.

Figure 7.4 presents the reference and the measured (estimated) motor speed, phase current and torque component current corresponding to the **BLAC** default settings. If the user will load **BLAC** and execute the motion, after up-loading the motion results, similar time-variation of these signals must be obtained.

Once the system operation is tested, different reference shapes and controller parameters may be tested.

Remarks:

1. Do not use for the current loop sampling period values below 40 μ s, which represents the maximum length of the current control interrupt routine.

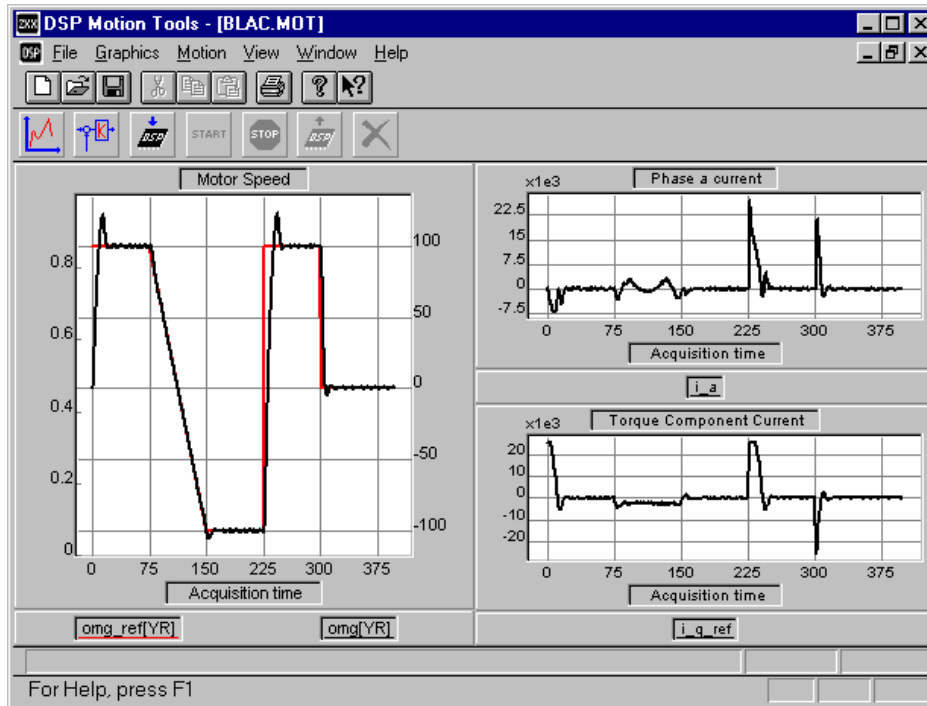


Figure 7.4. BLAC default motion results

2. The speed and current control sampling rates are completely independent and may be varied separately.
3. Be aware that when changing any of the controllers parameters (control coefficients or sampling rate value), one should re-tune the controller in order to get a satisfactory behavior of the motion system.

7.2.4 Other BLAC examples

Besides the **BLAC** application dedicated to standard configuration of the starter kit (drive and motor), the software also contains a **BLAC** application for the XBR-55-06-602-CE brushless motor. The application can be accessed by opening DSPMOT program and choose the **Blac_XBR.mot** project from ...:\MCWIN2833x_PRO\Demos\blac_XBR\.. The following files are included in the demo structure:

- **Blac_XBR.out**: the COFF format executable application file
- **Blac_XBR.map**: the MAP file of the application
- **Blac_XBR.mot**: the **DSPMOT** project associated to the application

Remark: The XBR-55-06-602-CE brushless motor must be connected to the ACMP750 3-phase GBT power module.

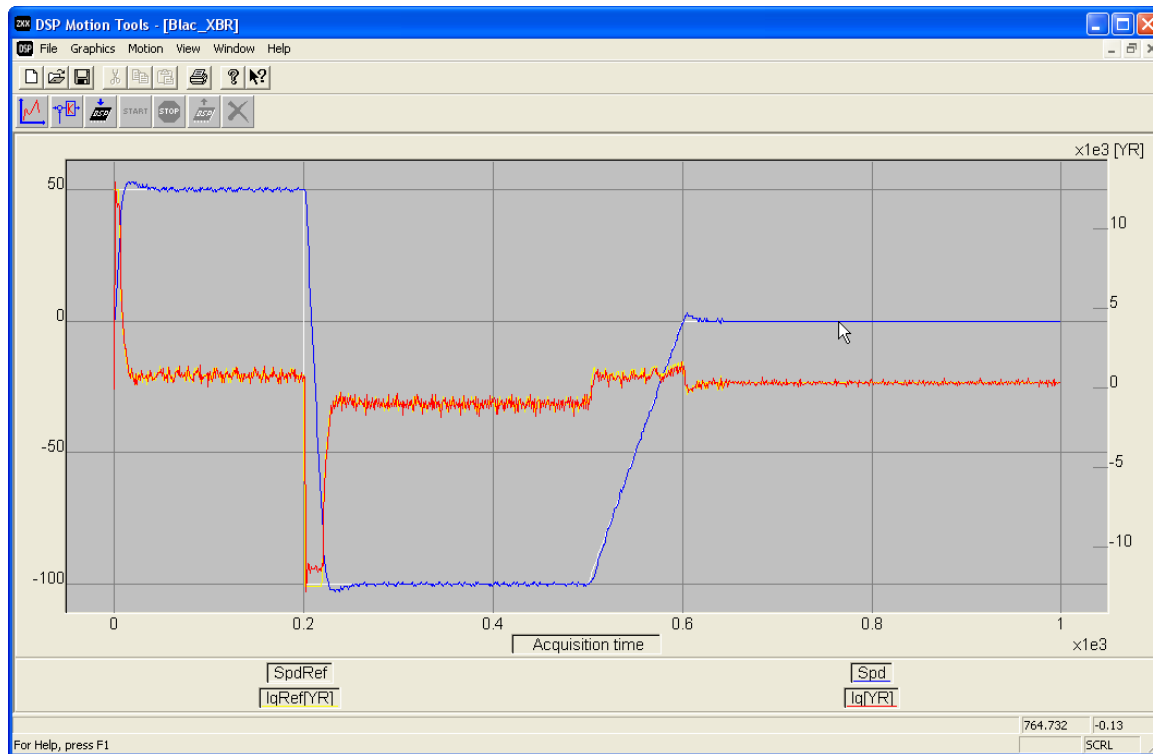


Figure 7.5. BLAC default motion results for the XBR-55-06-602-CE

Figure 7.5 presents the reference and the measured (estimated) motor speed and torque component current corresponding to the **BLAC** default settings. If the user will load **BLAC** and execute the motion, after up-loading the motion results, similar time-variation of these signals must be obtained.

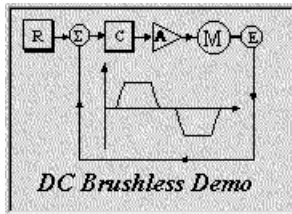
Once the system operation is tested, different reference shapes and controller parameters may be tested.

7.3 Brushless DC motion demo application

The brushless DC motion demo application implements a block commutation control method to drive the motor included in the **MCK28335** kit.

The demo is supplied as a TMS320F28335 application, included in a *.MOT project of the **DSPMOT** program. The following files are included in the demo structure:

- **bldc.out**: the COFF format executable application file
- **bldc.map**: the MAP file of the application
- **bldc.mot**: the **DSPMOT** project associated to the application



A mouse click on the « **Brushless DC demo** » button will activate the **BLDC** motion demo. Executed in the **DSPMOT** program environment, this example implements a speed control loop for a three phase brushless motor, driven in DC mode. The user may define the **PI** discrete current and speed controllers parameters, as well as the speed reference shape. At an advanced level, the user may include his own controllers and/or reference functions in the **BLDC** application and replace the demos ones.

7.3.1 Basic structure of the control scheme for the BLDC application

The **BLDC** application control scheme is presented in Figure 7.5. As one can see, the scheme is based on the measure of two phase currents and of the motor position. The speed estimator block is, as previously stated, a simple difference block. The measured phase currents, i_a and i_b , are used to compute the equivalent DC current in the motor, based on the Hall sensors position information. Remark that the Hall sensors give a 60 electrical degrees position information. The speed and current controllers are, as stated before, **PI** discrete controllers. Only one current controller is needed in this case, similar to a DC motor case. The voltage commutator block implements (by software) the computation of the phase voltages references, v_{as}^* , v_{bs}^* and v_{cs}^* , applied to the inverter. Practically, the 6 full compare PWM outputs of the DSC controller are directly driven by the program, based on these reference voltages. In the **BLDC** case, only four of the inverter transistors are controlled for a given position of the motor. The scheme will commute to a specific command configuration, for each of the 60 degrees position sectors, based on the information read from the Hall sensors.

The speed reference signal is obtained using the reference generator of **DSPMOT**, included in the application (see par. 6.3.3 for details about this module of **DSPMOT**). Thus, the speed reference may be imposed at Windows level, from **DSPMOT**.

Also the controllers parameters are set in **DSPMOT**, at Windows level, from the **Motion | Setup Controller** menu command. The proportional and integral control factors, as well as the sampling periods for the current and speed control loops, are set using this **DSPMOT** command (see par. 6.3.4 for details about the control module and functions).

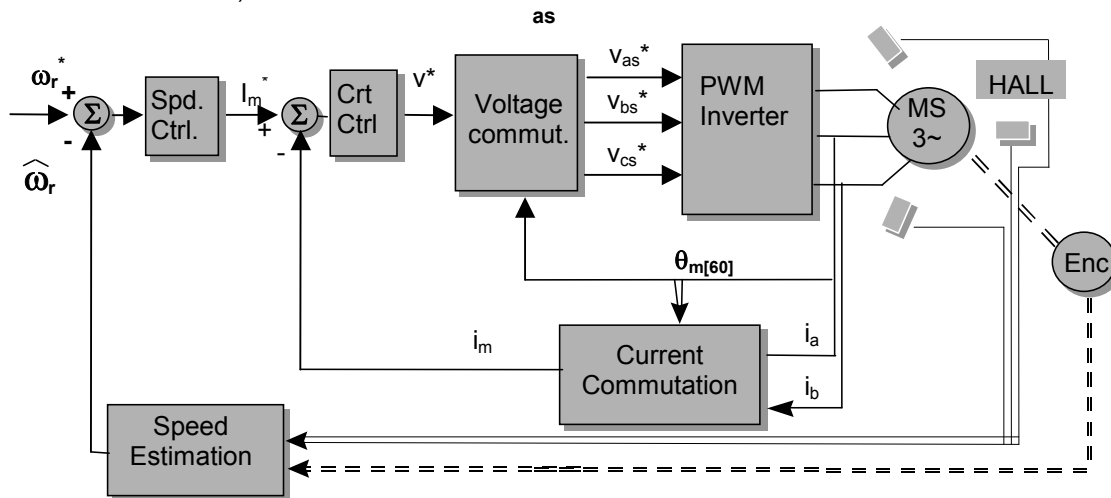


Figure 7.5. BLDC control scheme

7.3.2 BLDC demo parameters and variables

The **BLDC** application has the following control parameters, tuned for the motor included in the **MCK28335** kit:

current controllers:

	Pittman motor	Escap motor	Technosoft MBE.300E500 motor
Proportional factor - K_P	0.6	1.2	0
Integral factor - K_I	0.5	1.2	0.473
Sampling period - h_{crt}	0.1 ms	0.1 ms	0.1 ms

speed controller:

	Pittman motor	Escap motor	Technosoft MBE.300E500 motor
Proportional factor - K_P	160	120	155
Integral factor - K_I	10	2.5	19.4
Sampling period - h_{crt}	1 ms	1 ms	1 ms

Please note that these parameters are given at **DSPMOT** program level. **DSPMOT** will automatically scale the control coefficients values in order to convert them in the **IQ16** fractional representation (for more details about **IQ16** format see **IQmath library** documentation from Texas Instruments). The sampling periods values are used to compute the compare and period values for Timer 1 of the DSC controller, considered supplied with a 30 MHz clock signal.

During the tests which may be performed with the **BLDC** application, the user may select, at the **DSPMOT** level, different variables of the program, in order to store them during the motion, up-load and visualize them using the graphical features of **DSPMOT**. The file **BLDC.MAP** contains the complete list of these variables, which may be selected using the **Motion | Select DSC Variables** menu command of **DSPMOT**.

We present here the significance of the most important parameters and variables which may be considered by the user for motion evaluation purposes:

Motion parameters:

Name	Type	Description
IqController.Ki	IQ16	Integral coefficient for iq current controller (scaled)
SpeedController.Ki	IQ16	Integral coefficient for speed controller (scaled)
IqController.Kp	IQ16	Proportional coefficient for iq current controller (scaled)
SpeedController.Kp	IQ16	Proportional coefficient for speed controller (scaled)
_ctrl_crt_per	unsigned int	Current control period value
_ctrl_ps_per	unsigned int	Speed control period value

Motion variables:

Name	Type	Description
Hall.Hall	int	variable combining the Hall sensors outputs
Adc.Ia	IQ16	phase a current
Adc.Ib	IQ16	phase b current
Tabc2dq.IqRef	IQ16	rotor frame quadrature current component reference
Tabc2dq.Iq	IQ16	rotor frame quadrature current component
Adc.OffsetIa	long	offset measured on the A/D channel 6 (used for phase a current measurement)
Adc.OffsetIb	long	offset measured on the A/D channel 5 (used for phase b current measurement)
SpeedController.Ref	IQ16	motor speed reference
SpeedEstimator.Speed	int	motor speed (estimated from position variation)
Encoder.Position	int	motor position
pwm.UaRef	IQ16	phase a reference voltage
pwm.UbRef	IQ16	phase b reference voltage
pwm.UcRef	IQ16	phase c reference voltage
IqController.Out	IQ16	rotor frame quadrature voltage reference

7.3.3 A BLDC example

The **BLDC** application has a predefined motion profile and parameters, which are automatically loaded at the start of the demo.

A similar behavior as in figure 7.4 is obtained for the reference and the measured (estimated) motor speed, phase current and torque component current corresponding to the **BLDC** default settings. If the user will load **BLDC** and execute the motion, after up-loading the motion results, similar time-variation of these signals must be obtained.

Once the system operation is tested, different reference shapes and controller parameters may be tested.

Notes:	1). Do not use for the current loop sampling period values below 40 μ s, which represents the maximum length of the current control interrupt routine.
	2). The speed and current control sampling rates are completely independent and may be varied separately.
	3). Be aware that when changing any of the controllers parameters (control coefficients or sampling rate value), one should re-tune the controller in order to get a satisfactory behavior of the motion system.
	4). Because the speed estimate was implemented (for simplicity considerations) using the position measurement from the incremental encoder, be aware that the BLDC demo need BOTH Hall AND quadrature encoder connected to the MSK28335 board! The test does not run properly if only the Hall sensor connection is used!

7.3.4 Other BLDC examples

Besides the **BLDC** application dedicated to standard configuration of the starter kit (drive and motor), the software also contains a **BLDC** application for the XBR-55-06-602-CE brushless motor. The application can be accessed by opening DSPMOT program and choose the **Bldc_XBR.mot** project from ...:\MCWIN2833x_PRO\Demos\ bldc_XBR\. The following files are included in the demo structure:

- **Bldc_XBR.out**: the COFF format executable application file
- **Bldc_XBR.map**: the MAP file of the application
- **Bldc_XBR.mot**: the **DSPMOT** project associated to the application

Remark: The XBR-55-06-602-CE brushless motor must be connected to the ACMP750 3-phase GBT power module.

A similar behavior as in figure 7.5 is obtained for the reference and the measured (estimated) motor speed, phase current and torque component current corresponding to the **BLDC** default settings. If the user will load **BLDC** and execute the motion, after up-loading the motion results, similar time-variation of these signals must be obtained.

Once the system operation is tested, different reference shapes and controller parameters may be tested.

7.4 Induction Motor Vector Control motion demo application

The Induction Motor Vector Control (**IMVC**) motion application implements a vector control method to drive a three-phase induction motor.

The demo is supplied as a TMS320F28335 application, included in a *.**MOT** project of the **DSPMOT** program. The application is a speed control application of the induction motor. The following files are included in the demo structure:

- **imvc.out**: the COFF format executable application file
- **imvc.map**: the MAP file of the application
- **imvc.mot**: the **DSPMOT** project associated to the application

The application can be accessed by opening DSPMOT program and choose the **imvc.mot** project from\MCWIN2833x_PRO\Demos\imvc \.

Executed in the **DSPMOT** program environment, this example implements a speed control loop for a induction motor, driven in vector control mode. The user may define the **PI** discrete current (i_d and i_q) and speed controllers parameters, as well as the speed reference shape. At an advanced level, the user may include his own controllers and/or reference functions in the **IMVC** application and replace the demo's ones.

7.4.1 Basic structure of the control scheme for the IMVC application

The **IMVC** application control scheme is presented in Figure 7.6. As one can see, the scheme is based on the measure of two phase currents and of the motor position. The Slip Compensation block is used to estimate the stator field position. The measured phase currents, i_a and i_b , are transformed into the stator reference frame components, i_{ds} and i_{qs} . Then, based on the position information, these components are transformed into the rotor frame direct and quadrature components, i_{de} and i_{qe} . The speed and current controllers are, as stated before, **PI** discrete controllers. The voltage controlled oscillator block (VCO) implements (by software) the coordinates transformation and computation of the phase voltages references, \mathbf{v}_{as}^* , \mathbf{v}_{bs}^* and \mathbf{v}_{cs}^* , applied to the inverter. Practically, the 6 full compare PWM outputs of the DSP controller are directly driven by the program, based on these reference voltages.

The direct current component reference i_{de}^* is set to **I_D_VAL** value applied to the d-axis current controller. This current value is based on the system parameters and corresponds to the motion of the motor at a constant flux level.

The speed reference signal is obtained using the reference generator of **DSPMOT**, included in the application (see par. 6.3.3 for details about this module of **DSPMOT**). Thus, the speed reference may be imposed at Windows level, from **DSPMOT**.

Also the controllers parameters are set in **DSPMOT**, at Windows level, from the **Motion | Setup Controller** menu command. The proportional and integral control factors, as well as the sampling periods for the current and speed control loops, are set using this **DSPMOT** command (see par. 6.3.4 for details about the control module and functions).

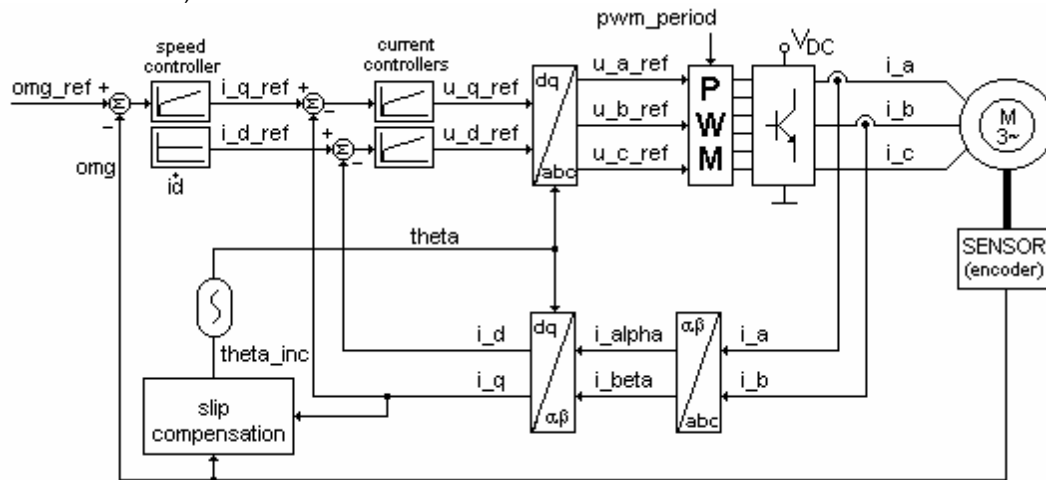


Figure 7.6 IMVC application control scheme

7.4.2 IMVC demo parameters and variables

The **IMVC** application has the following control parameters, tuned for the motor included in the **MCK28335** kit:

current controllers:

	Technosoft ASPMN00011 motor
Proportional factor - K_P	1.3
Integral factor - K_I	0.28
Sampling period - h_{crt}	0.1 ms

speed controller:

	Technosoft ASPMN00011 motor
Proportional factor - K_P	1357
Integral factor - K_I	242
Sampling period - h_{crt}	1 ms

Please note that these parameters are given at **DSPMOT** program level. **DSPMOT** will automatically scale the control coefficients values in order to convert them in the **IQ16** fractional representation (for more details about **IQ16** format see **IQmath library** documentation from Texas Instruments). The sampling periods values are used to compute the compare and period values for Timer 1 of the DSP controller, if supplied with a 30MHz clock signal.

During the tests which may be performed with the **IMVC** application, the user may select, at the **DSPMOT** level, different variables of the program, in order to store them during the motion, up-load and visualize them using the graphical features of **DSPMOT**. The file **IMVC.MAP** contains the complete list of these variables, which may be selected using the **Motion | Select DSP Variables** menu command of **DSPMOT**.

We present here the significance of the most important parameters and variables which may be considered by the user for motion evaluation purposes:

Motion parameters:

Name	Type	Description
IdController.Ki	IQ16	Integral coefficient for id current controller (scaled)
IqController.Ki	IQ16	Integral coefficient for iq current controller (scaled)
SpeedController.Ki	IQ16	Integral coefficient for speed controller (scaled)
IdController.Kp	IQ16	Proportional coefficient for id current controller (scaled)
IqController.Kp	IQ16	Proportional coefficient for iq current controller (scaled)
SpeedController.Kp	IQ16	Proportional coefficient for speed controller (scaled)
_ctrl_crt_per	unsigned int	Current control period value

_ctrl_ps_per	unsigned int	Speed control period value
--------------	-----------------	----------------------------

Motion variables:

Name	Type	Description
ThetaEstimator.ElectricPosition	int	electric position angle in encoder pulses, scaled for an electric period
ThetaEstimator.Theta	IQ16	electric position angle in Q15 format
Encoder.Position	int	motor position
Adc.Ia	IQ16	phase a current
Adc.Ib	IQ16	phase b current
Tabc2dq.Id	IQ16	rotor frame direct current component
Tabc2dq.IdRef	IQ16	rotor frame direct current component reference
Tabc2dq.IqRef	IQ16	rotor frame quadrature current component reference
Tabc2dq.Iq	IQ16	rotor frame quadrature current component
Adc.OffsetIa	long	offset measured on the A/D channel 6 (used for phase a current measurement)
Adc.OffsetIb	long	offset measured on the A/D channel 5 (used for phase b current measurement)
SpeedController.Ref	IQ16	motor speed reference
SpeedEstimator.Speed	int	motor speed (estimated from position variation)
pwm.UaRef	IQ16	phase a reference voltage
pwm.UbRef	IQ16	phase b reference voltage
pwm.UcRef	IQ16	phase c reference voltage
IdController.Out	IQ16	rotor frame direct voltage reference
IqController.Out	IQ16	rotor frame quadrature voltage reference

7.4.3 An IMVC example

The **IMVC** application has a predefined motion profile and parameters, which are automatically loaded at the start of the demo.

Figure 7.7 presents the reference and the measured (estimated) motor speed, phase current and torque component current corresponding to the **IMVC** default settings. If the user will load **IMVC** and execute the motion, after up-loading the motion results, similar time-variation of these signals must be obtained.

Once the system operation is tested, different reference shapes and controller parameters may be tested.

Remarks:

- Do not use for the current loop sampling period values below 40 μ s, which represents the maximum length of the current control interrupt routine.

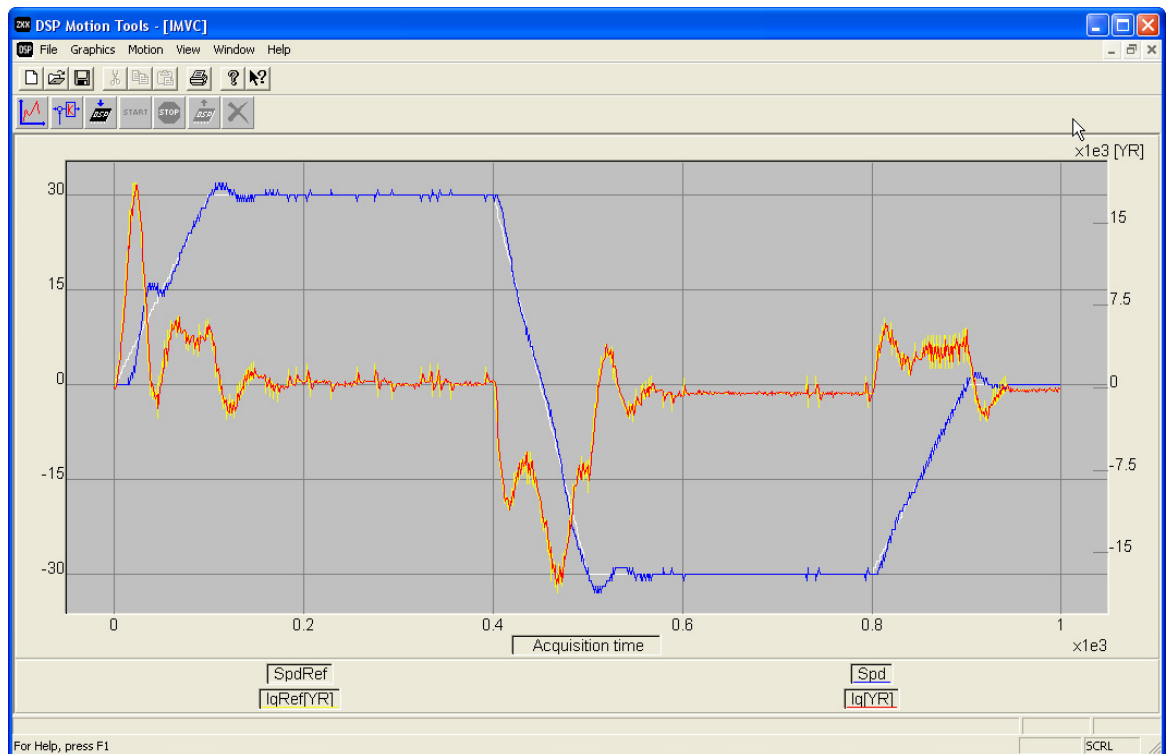


Figure 7.6. IMVC default motion results

5. The speed and current control sampling rates are completely independent and may be varied separately.
6. Be aware that when changing any of the controllers parameters (control coefficients or sampling rate value), one should re-tune the controller in order to get a satisfactory behavior of the motion system.

Remark: The ASPMN00011 induction motor must be connected to the ACMP750 3-phase GBT power module.

This page is empty

8 DMC28x Developer Demos

This chapter describes the structure folders for **DMC28x Developer** demos included in all **MSWIN28335 / MCWIN28335** platforms.

Contents

- 8.1. Assembly demos
- 8.2. C/C++ demos
- 8.3. How to run a demo

8.1 Assembly demos

The folder <MxWIN28335 folder>\ASMDemos contains demos written in 'F28x assembly language.

These projects can be open / build / run with **DMC28x Developer Lite** from **MCWIN28335 / MSWIN28335** platforms or **DMC28x Developer Pro** from **MCWIN28335 Pro / MSWIN28335 Pro** platforms.

Project Folder	Demo	Description
<MxWIN28335 folder>\ASMDemos\	-	This folder contains the headers files ("F28335_Regs_a.h" and "F28335_Regs_glob_a.h") included in demos and the demos folders

8.2 C/C++ demos

The folder <MxWIN28335_folder>\CDemos contains demos written in C/C++ language.

These projects can be open / build / run just with **DMC28x Developer Pro** from **MCWIN28335 Pro / MSWIN28335 Pro** platforms. The **DMC28x Developer Lite** version cannot open these projects.

The **DMC28x Developer Pro** version can build both sources types (C/C++ and assembly) and can use compiling tools of **Technosoft** and **Texas Instruments**.

Project Folder	Demo	Description
<MxWIN28335 folder>\CDemos\	-	This folder contains the headers files ("LF28335_Regs.h" and "RegDef.h") included in demos and the demos folders
<MxWIN28335 folder>\CDemos\Adc	ADC demo	This application shows how to perform A/D conversion using the on-chip 12-bit AD converter module. This has 16 analog inputs, and accepts to command SOC for 2 channels at once. In this example channels ADCIN0 and ADCIN1 are used. The start of conversion (SOC) is done automatically by underflow of ePWM1 timer. The ADC ISR is used to store results in a buffer of 2 x 128 words length.
<MxWIN28335 folder>\CDemos\Can	CAN demo	This application shows how to program the CAN-bus interface in selftest mode. Data to send (8 bytes) is setup in variables: tx_data_val[0...4]
<MxWIN28335 folder>\CDemos\Cap	Capture demo	The application shows how to program captures on each one of the 6 existing capture module. This example programs the capture module eCAP1 to detect rising edges and to capture timer value in their CAP1..4 registers.
<MxWIN28335 folder>\CDemos\Dac	DAC demo	This demo exemplifies how to configure the AD5322 DAC converter into the following modes:

		<ul style="list-style-type: none"> - update immediate on the two outputs - simultaneously update on the two outputs.
<MxWIN28335 folder>\CDemos\E2rom	E2ROM demo	This demo exemplifies how to read/write 8 consecutive memory locations from/into the SPI E2ROM memory.
<MxWIN28335 folder>\CDemos\Gpio	General-purpose I/O's demo	This application shows how to use the General Purpose I/Os. It also shows bit-manipulation techniques since I/Os from port A.
<MxWIN28335 folder>\CDemos\Gpt_1	General Purpose Timers demo 1	This application shows some of the features (counting mode, compare/PWM output generation) offered by the ePWM timers. The ePWM module selected for this application is ePWM1. It is configured in up-count mode with prescale ratio 1. The compare output of the ePWM module is enabled to be active low when a compare match occurs.
<MxWIN28335 folder>\CDemos\Gpt_2	General Purpose Timers demo 2	<p>This application shows some of the possibilities offered by the timers of ePWM modules to generate interrupts to the processor. The ePWM module selected for this application is ePWM1. It is configured in up-count mode with prescale ratio 1. The compare output of the ePWM module is enabled to be active low when a compare match occurs.</p> <p>The ePWM module interrupt is enabled. A test variable count_periodes is incremented in the corresponding interrupt service routines.</p>
<MxWIN28335 folder>\CDemos\i2c	I2C demo	This application programs the I2C to send/receive a 16 bit stream representing a character code entered by user in the variable code_to_send.
<MxWIN28335 folder>\CDemos\Log_ref	Logger and reference generator demo	The project implements an example for using of supplementary functionality - reference generator and a data logger function, usable with the DMCD-pro development platform of Technosoft and allowing a high-level interface with the program, from a PC Windows environment.
<MxWIN28335 folder>\CDemos\McBSP	McBSP demo	This application programs the McBSP to send/receive a 32 bit stream from the variable data_to_send.
<MxWIN28335 folder> \CDemos\Monitor \ExtCodeFlash	Stand-alone demo	This application is an example for automatically execution of the user code from flash memory and initialization of monitor communication.
<MxWIN28335 folder> \CDemos\Monitor \ExtendedMonitor	Extended Command Interpreter demo	The application is an example to show how to extend the command interpreter into the RAM memory.
<MxWIN28335 folder>\CDemos\Pwma	Asymmetric PWM demo	The application generates three-phase voltage using asymmetric pulse width modulation (PWM) technique. Timers and compare units of ePWM1, ePWM2, ePWM3 modules are programmed to give 6 PWM signals in PWM asymmetric mode. As a result, three line voltages are

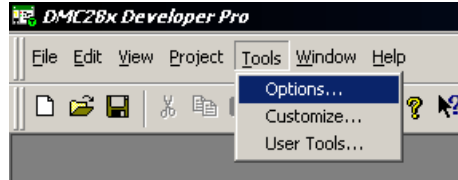
		generated at the output of the power inverter bridge.
<MxWIN28335 folder>\CDemos\Pwms	Symmetric PWM demo	The application generates three-phase voltage using symmetric pulse width modulation (PWM) technique. Timers and compare units of ePWM1, ePWM2, ePWM3 modules are programmed to give 6 PWM signals in PWM symmetric mode. As a result, three line voltages are generated at the output of the power inverter bridge.
<MxWIN28335 folder>\CDemos\Qep	QEP demo	The application enables Quadrature Encoder Pulse Circuit of eQEP1 to decode and count the quadrature encoded input pulses A and B on pins QEP1 and QEP2.
<MxWIN28335 folder>\CDemos\Spi	SPI demo	This application programs the SPI to send/receive a 16 bit stream representing a character code entered by user in the variable code_to_send.
<MxWIN28335 folder>\CDemos\Wd	Watchdog demo	The application consists in activating General Purpose Timer 2 period interrupt routine in which a counter variable is increased and the WD control registers are updated. If the Timer 2 interrupt period is larger than the watchdog period or the watchdog updated key values are wrong, the module generates a processor reset.

8.3 How to run a demo

8.3.1 DMC28x Developer Pro settings

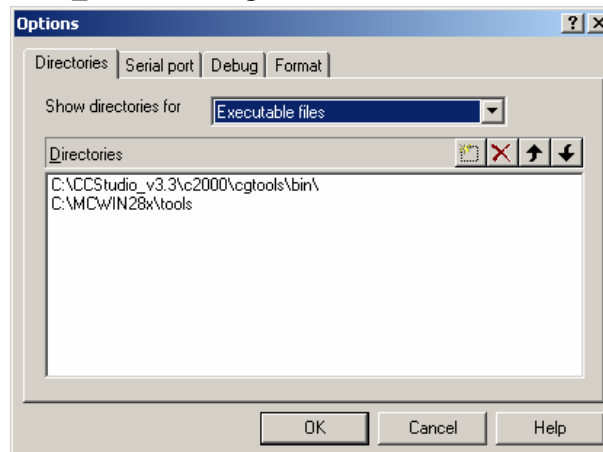
In order to build a C program under **DMC28x Developer Pro** the following settings must be performed:

1. If you don't have the TI C-compiling tools installed, please install it. The C-compiling tools are included into the Code Composer package.
2. Open the **DMC28x Developer Pro** program (included in **MSWIN28335 Pro / MCWIN28335 Pro** packages) and select **Tools | Options** menu command.



3. In the **Directories** tab, when **Executable files** at **Show directories for** combo box, check if the TI C-compiling tools path is set correctly. If the default installation path of Code Composer was kept, this path is:

C:\CCStudio_v3.3\c2000\cgtools\bin

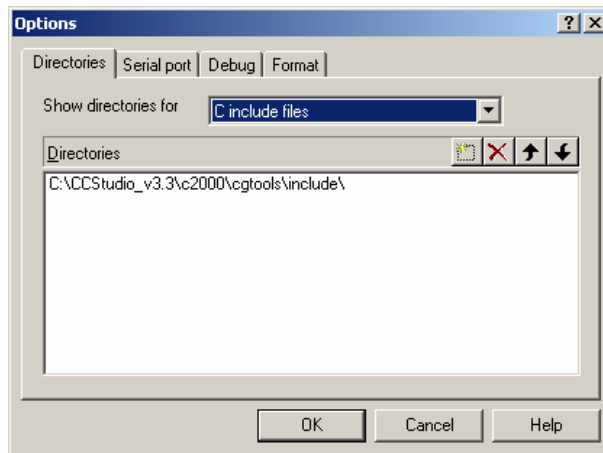


For older versions of Code Composer the path could be: **C:\CCStudio_v3.1\cgtools\bin**
or **C:\ti\c2000\cgtools\bin**

Please set this path according to the Code Composer installation folder.

4. Select **C include files** at **Show directories for** combo box and check if the TI C-compiling tools path is set correctly. If the default installation path of Code Composer was kept, this path is:

C:\CCStudio_v3.3\c2000\cgtools\include

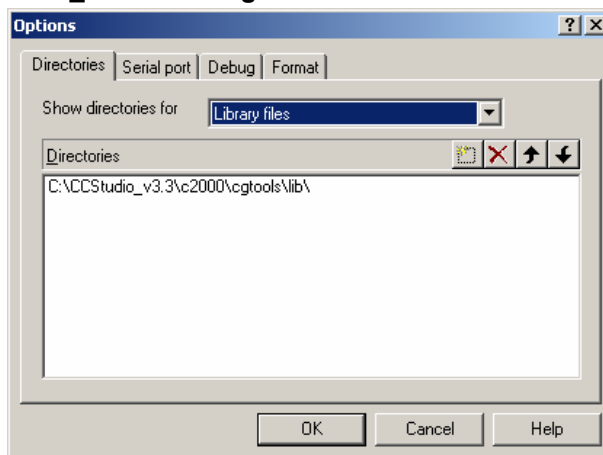


For older versions of Code Composer the path could be: **C:\CCStudio_v3.1\cgtools\include**
or **C:\ti\c2000\cgtools\include**

Please set this path according to the Code Composer installation folder.

5. Select **Library files** at **Show directories for** combo box and check if the TI C-compiling tools path is set correctly. If the default installation path of Code Composer was kept, this path is:

C:\CCStudio_v3.3\c2000\cgtools\lib



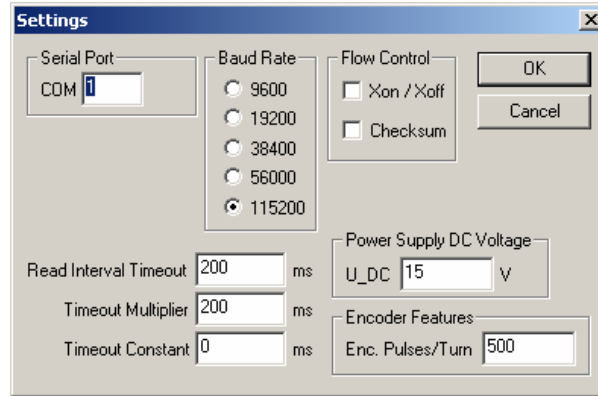
For older versions of Code Composer the path could be: **C:\CCStudio_v3.1\cgtools\lib**
or **C:\ti\c2000\cgtools\lib**

Please set this path according to the Code Composer installation folder.

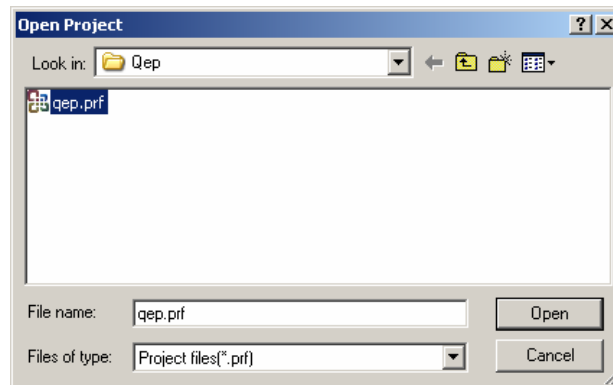
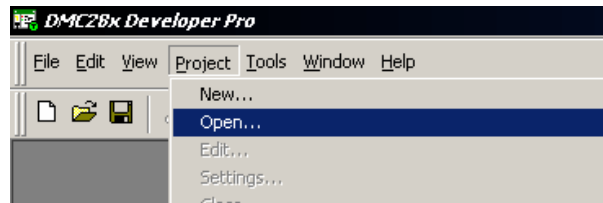
8.3.2 Demo running

This is an example for the execution of QEP demo (<MXWIN28x folder>\CDemos\Qep\) under **DMC28x Developer Pro** platform. Follow steps:

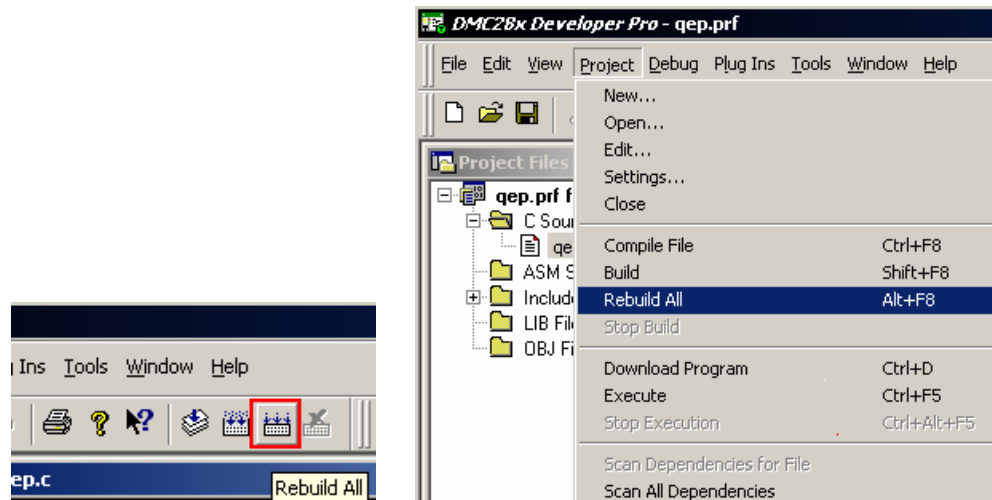
1. Make the H/W connections (serial cable, supply, etc.) and power-on the board
2. Select the COM port in **MxWIN28x -> Settings** dialog and test the communication



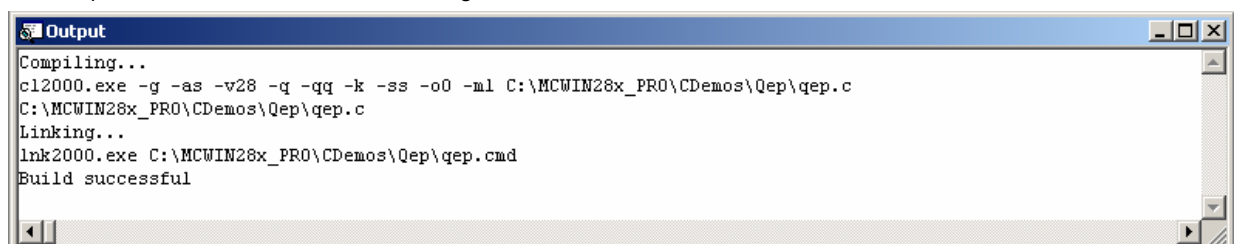
3. Select **Project | Open** menu command



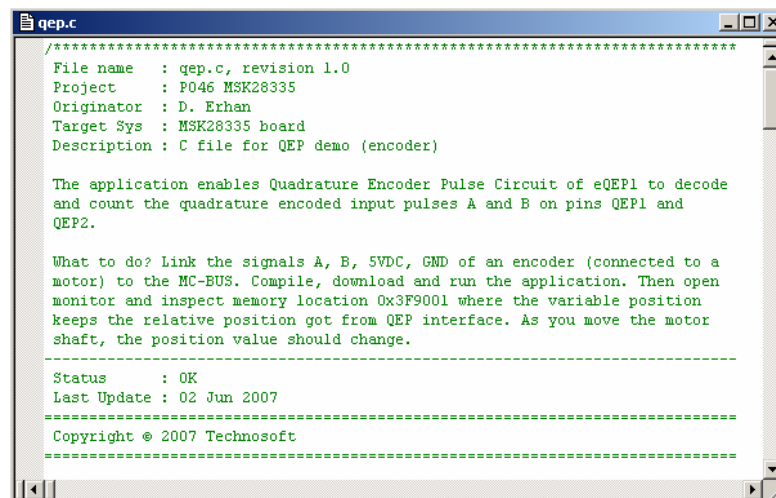
4. Rebuild project pressing the **Rebuild All** button or selecting **Project | Rebuild All** menu command



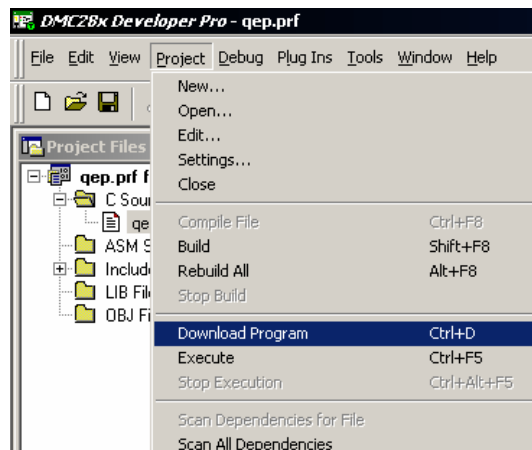
The output window after successful building



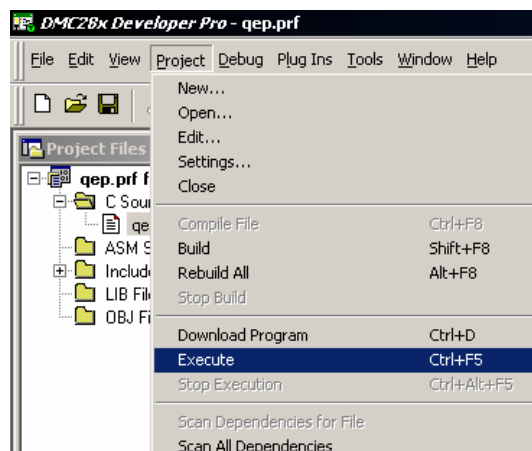
5. Open the demo source file and follow the instructions specified in the commented text at the beginning of the file.



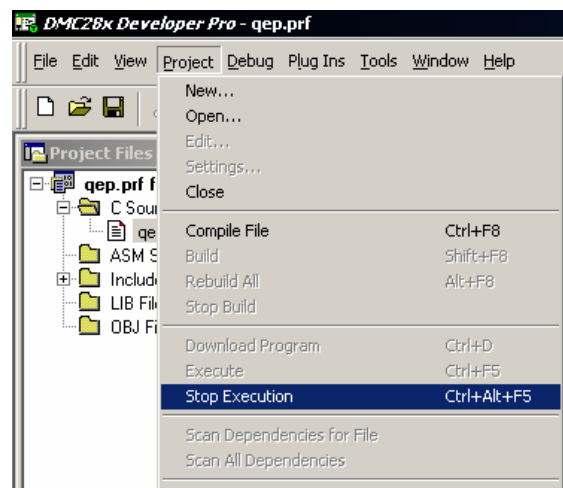
6. Select **Project | Download Program** menu command to download the program



7. Select **Project | Execute** menu command to execute the program



8. Check the demo results according to the instructions from source file (step 5)
9. Select **Project | Stop Execution** menu command to stop the program and reset the DSP



APPENDICES

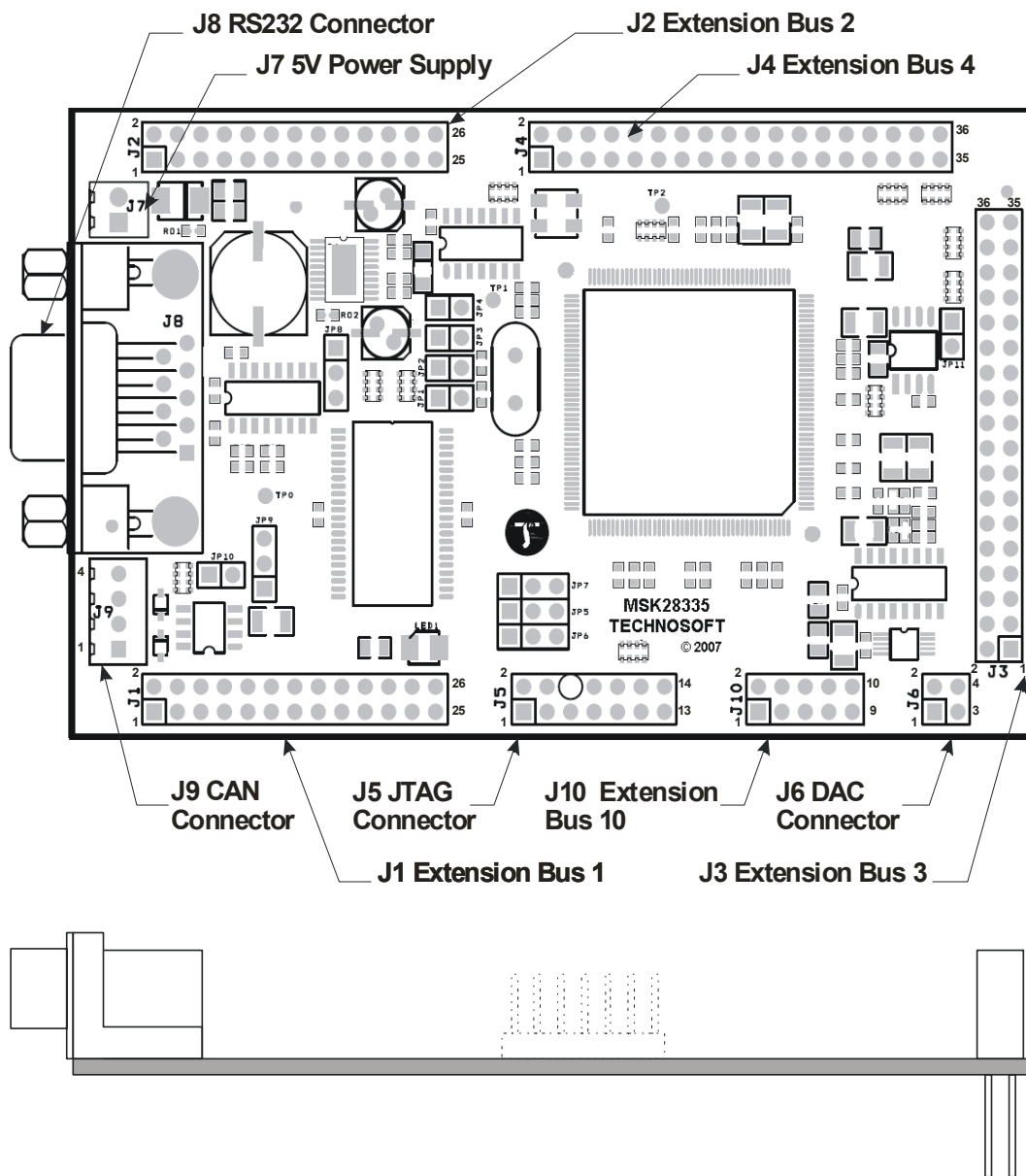
MSK28335 / MCK28335

Technical Specifications

This page is empty

Appendix A. MSK28335 DSC board - Technical data

A.1. MSK28335 DSC board - Connectors



This drawing is not to scale 1:1

J1 – Extension Bus 1

Pin	Signal	Description
1	GPIO79 / XD0	Digital DSC signal
2	GPIO71 / XD8	Digital DSC signal
3	GPIO78 / XD1	Digital DSC signal
4	GPIO70 / XD9	Digital DSC signal
5	GPIO77 / XD2	Digital DSC signal
6	GPIO69 / XD10	Digital DSC signal
7	GPIO76 / XD3	Digital DSC signal
8	GPIO68 / XD11	Digital DSC signal
9	GPIO75 / XD4	Digital DSC signal
10	GPIO67 / XD12	Digital DSC signal
11	GPIO74 / XD5	Digital DSC signal
12	GPIO66 / XD13	Digital DSC signal
13	GPIO73 / XD6	Digital DSC signal
14	GPIO65 / XD14	Digital DSC signal
15	GPIO72 / XD7	Digital DSC signal
16	GPIO64 / XD15	Digital DSC signal
17	GPIO36 / SCIRXDA / XZCS0	Digital DSC signal
18	XRD	Digital DSC signal
19	GPIO28 / SCIRXDA / XZCS6	Digital DSC signal
20	+3.3V	+3.3VDC Supply Output
21	GPIO37 / ECAP2 / XZCS7	Digital DSC signal
22	GPIO38 / XWE0	Digital DSC signal
23	GPIO35 / SCITXDA / XR / W	Digital DSC signal
24	GND	Digital ground
25	GPIO15 / TZ4 / XHOLDA / SCIRXDB / MFSXB	Digital DSC signal
26	GPIO14 / TZ3 / XHOLD / SCITXDB / MCLKXB	Digital DSC signal

J2 – Extension Bus 2

Pin	Signal	Description
1	GPIO40 / XA0 / XWE1	Digital DSC signal
2	GPIO80 / XA8	Digital DSC signal
3	GPIO41 / XA1	Digital DSC signal
4	GPIO81 / XA9	Digital DSC signal
5	GPIO42 / XA2	Digital DSC signal
6	GPIO82 / XA10	Digital DSC signal
7	GPIO43 / XA3	Digital DSC signal
8	GPIO83 / XA11	Digital DSC signal
9	GPIO44 / XA4	Digital DSC signal

10	GPIO84 / XA12	Digital DSC signal
11	GPIO45 / XA5	Digital DSC signal
12	GPIO85 / XA13	Digital DSC signal
13	GPIO46 / XA6	Digital DSC signal
14	GPIO86 / XA14	Digital DSC signal
15	GPIO47 / XA7	Digital DSC signal
16	GPIO87 / XA15	Digital DSC signal
17	GPIO34 / ECAP1 / XREADY	Digital DSC signal
18	#RSTOUT	Reset output
19	XCLKOUT	Digital DSC signal
20	XRS	Digital DSC signal
21	+5V	+5V DC Supply Input or Output
22	+3.3V	+3.3VDC Supply Output
23	GPIO39 / XA16	Digital DSC signal
24	GPIO30 / CANRXA / XA18	Digital DSC signal
25	GPIO31 / CANTXA / XA17	Digital DSC signal
26	GPIO29 / SCITXDA / XA19	Digital DSC signal

J3 – Extension Bus 3

Pin	Signal	Description
1	+3.3V	+3.3V _{DC} Supply Output
2	+3.3V	+3.3V _{DC} Supply Output
3	GPIO0 / EPWM1A	Digital DSC signal
4	GPIO1 / EPWM1B / ECAP6 / MFSRB	Digital DSC signal
5	GPIO2 / EPWM2A	Digital DSC signal
6	GPIO3 / EPWM2B / ECAP5 / MCLKRB	Digital DSC signal
7	GPIO4 / EPWM3A	Digital DSC signal
8	GPIO5 / EPWM3B / MFSRA / ECAP1	Digital DSC signal
9	GPIO48 / ECAP5 / XD31	Digital DSC signal
10	GPIO49 / ECAP6 / XD30	Digital DSC signal
11	GPIO32 / SDAA / EPWMSYNCl / ADCSOCAO	Digital DSC signal
12	GPIO33 / SCLA / EPWMSYNCO / ADCSOCBO	Digital DSC signal
13	GPIO50 / EQEP1A / XD29	Digital DSC signal
14	GPIO51 / EQEP1B / XD28	Digital DSC signal
15	GPIO53 / EQEP1I / XD26	Digital DSC signal
16	GPIO12 / CANTXB / MDXB / TZ1	Digital DSC signal
17	GPIO52 / EQEP1S / XD27	Digital DSC signal
18	GPIO13 / TZ2 / CANRXB / MDRB	Digital DSC signal
19	GPIO54 / SPISIMOA / XD25	Digital DSC signal
20	GPIO55 / SPISOMIA / XD24	Digital DSC signal
21	GPIO56 / SPICLKA / XD23	Digital DSC signal
22	GPIO57 / SPISTEA / XD22	Digital DSC signal
23	GND	Digital ground
24	GND	Digital ground
25	+5V	+5V DC Supply Input or Output
26	+5V	+5V DC Supply Input or Output
27	+3VA	+3V analog reference
28	AGND	Analog ground
29	ADCINA0	Analog DSC signal
30	ADCINA1	Analog DSC signal
31	ADCINA2	Analog DSC signal
32	ADCINA3	Analog DSC signal
33	ADCINA4	Analog DSC signal
34	ADCINA5	Analog DSC signal
35	ADCINA6	Analog DSC signal
36	AGND	Analog ground

J4 – Extension Bus 4

Pin	Signal	Description
1	+3.3V	+3.3VDC Supply Output
2	+3.3V	+3.3VDC Supply Output
3	GPIO6 / EPWM4A / EPWMSYNCI / EPWMSYNCO	Digital DSC signal
4	GPIO7 / EPWM4B / MCLKRA / ECAP2	Digital DSC signal
5	GPIO8 / EPWM5A / CANTXB / ADCSOCAO	Digital DSC signal
6	GPIO9 / EPWM5B / SCITXDB / ECAP3	Digital DSC signal
7	GPIO10 / EPWM6A / CANRXB / ADCSOCBO	Digital DSC signal
8	GPIO11 / EPWM6B / SCIRXDB / ECAP4	Digital DSC signal
9	GPIO60 / MCLKRB / XD19	Digital DSC signal
10	GPIO61 / MFSRB / XD18	Digital DSC signal
11	GPIO63 / SCITXDC / XD16	Digital DSC signal
12	GPIO62 / SCIRXDC / XD17	Digital DSC signal
13	GPIO24 / ECAP1 / EQEP2A / MDXB	Digital DSC signal
14	GPIO25 / ECAP2 / EQEP2B / MDRB	Digital DSC signal
15	GPIO26 / ECAP3 / EQEP2I / MCLKXB	Digital DSC signal
16	GPIO16 / SPISIMOA / CANTXB / TZ5	Digital DSC signal
17	GPIO27 / ECAP4 / EQEP2S / MFSXB	Digital DSC signal
18	GPIO17 / SPISOMIA / CANRXB / TZ6	Digital DSC signal
19	SCITXA_MCBUS	Connected to GPIO29 / SCITXDA / XA19 or GPIO35 / SCITXDA / XR / W through JP7
20	SCIRXA_MCBUS	Connected to GPIO28 / SCIRXDA / XZCS6 or GPIO36 / SCIRXDA / XZCS0 through JP6, JP8
21	GPIO19 / SPISTEA / SCIRXDB / CANTXA	Digital DSC signal
22	CANRXA_MCBUS	Connected to GPIO18 / SPICLKA / SCITXDB / CANRXA through JP9
23	GND	Digital ground
24	GND	Digital ground
25	+5V	+5V DC Supply Input or Output
26	+5V	+5V DC Supply Input or Output
27	+3VA	+3V analog reference
28	AGND	Analog ground
29	ADCINB0	Analog DSC signal
30	ADCINB1	Analog DSC signal
31	ADCINB2	Analog DSC signal
32	ADCINB3	Analog DSC signal
33	ADCINB4	Analog DSC signal
34	ADCINB5	Analog DSC signal
35	ADCINB6	Analog DSC signal
36	ADCREFIN_BUFF	Analog DSC signal, buffered

J10 – Extension Bus 10

Pin	Signal	Description
1	GPIO20 / EQEP1A / MDXA / CANTXB	Digital DSC signal
2	GPIO21 / EQEP1B / MDRA / CANRXB	Digital DSC signal
3	GPIO22 / EQEP1S / MCLKXA / SCITXDB	Digital DSC signal
4	GPIO58 / MCLKRA / XD21	Digital DSC signal
5	GPIO23 / EQEP1I / MFSXA / SCIRXDB	Digital DSC signal
6	GPIO59 / MFSRA / XD20	Digital DSC signal
7	GND	Digital ground
8	+3.3V	+3.3V DC Supply Output

J5 – JTAG

Pin	Signal / 'F28335 Pin	Pin	Signal / 'F28335 Pin
1	TMS	2	TRST
3	TDI	4	GND
5	+3.3V	6	No pin (key)
7	TDO	8	GND
9	TCK	10	GND
11	TCK	12	GND
13	EMU0	14	EMU1

J6 – DAC

Pin	Signal	Pin	Signal
1	12-bit DAC1 output in range 0-3V	2	VREFLO (analog GND)
3	12-bit DAC2 output in range 0-3V	4	DACREF – DAC reference voltage (3V output - default).

Note:

1. Default output range for DAC1 and DAC2 is 0-3V.
2. The DAC used on MSK28335 DSC board is AD5322 from Analog Devices.

J7 – Power supply

Pin	Signal
1	+5V _{DC} , minimum 500mA
2	GND (supply return)

J8 – RS-232

Pin	Signal	Pin	Signal
1	Not connected	6	Not connected
2	TxD	7	Not connected
3	RxD	8	Not connected
4	Not connected	9	Not connected
5	GND		

Note: The **MSK28335 DSC board** has a DB9 female connector. For RS-232 serial communication with a PC, use a straight-through serial cable having a DB9 male connector on the MSK28335 DSC board side and a DB9 female connector on the PC side. If your PC connector type is DB25 use a DB9 to DB25 adapter or use a serial cable with a DB25 female connector on the PC side. Since only 3 signals are used for RS-232 serial communication, **Fig. A.1** and **Fig. A.2** presents the minimal connections needed.

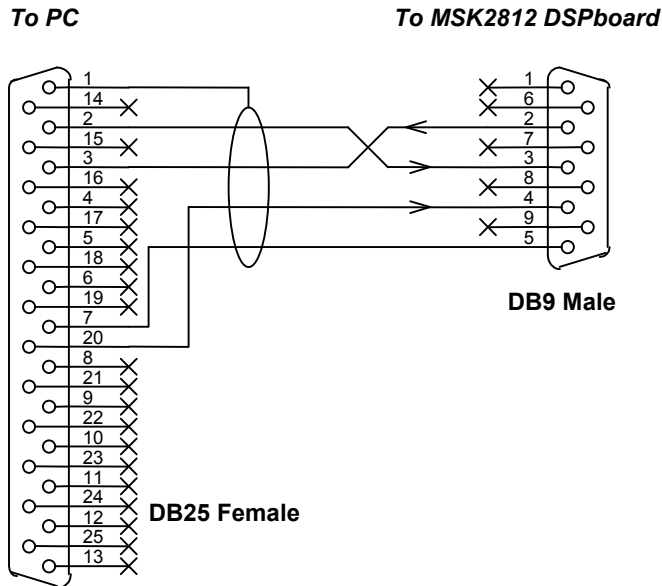


Figure A.1. Minimal connections for a DB9 to DB25 serial

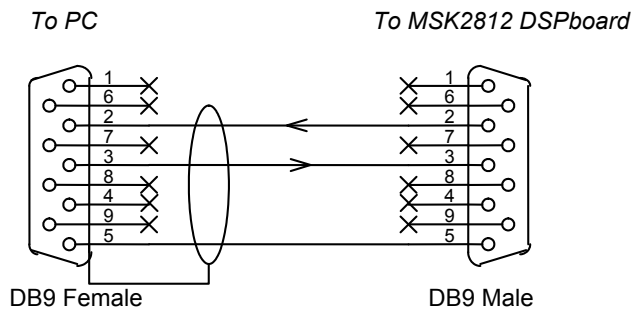


Figure A.2. Minimal connections for a DB9 to DB9 serial cable

J9 – CAN

Pin	Signal
1	CAN Supply output (+5V _{DC})
2	CAN-HI High data line
3	CAN-LO Low data line
4	CAN Ground

Notes:

1. The MSK28335 DSC board CAN interface is compliant with CAN-Bus v2.0B standard.
2. The connection of +5V_{DC} pin is need only if the other one device connected to MSK28335 board does not have the supply for CAN interface.
3. The CAN-bus network must include 2 terminating resistors of 120Ω each. Each MSK28335 DSC board has a 120Ω resistor, which can be connected across CAN-HI and CAN-LO lines by strapping the solder joint **JP9**. The default position of **JP9** is open (not mounted). The **JP9** must be strapped only on the MSK28335 DSC boards placed at the network boundary.
4. The CAN-Bus cable must fulfil the desired requirements, depending on the desired bit rate, as specified in the ISO-DIS 11898 standard. The most important parameters of the recommended cable are listed in **Table A.1**. An extract of the ISO-DIS 11898 cable length recommendations is presented in **Table A.2**.

Table A.1. ISO-DIS 11898 CAN-Bus cable parameters recommendations

Parameter	Recommended value
Cable type	Twisted pair, shielded or unshielded
Cable impedance	Typ. 62 ohm
Cable distributed delay	Typ. 5ns / meter

Table A.2. ISO-DIS 11898 Cable length recommendations

Bit rate	Recommended maximum bus length
1 Mbit / sec.	40m
800 Kbit / sec.	50m
500 Kbit / sec.	100m
250 Kbit / sec.	250m
125 Kbit / sec.	500m
50 Kbit / sec.	1000m
20 Kbit / sec.	2500m
10 Kbit / sec.	5000m

A.2. MSK28335 DSC board – Signals Connections

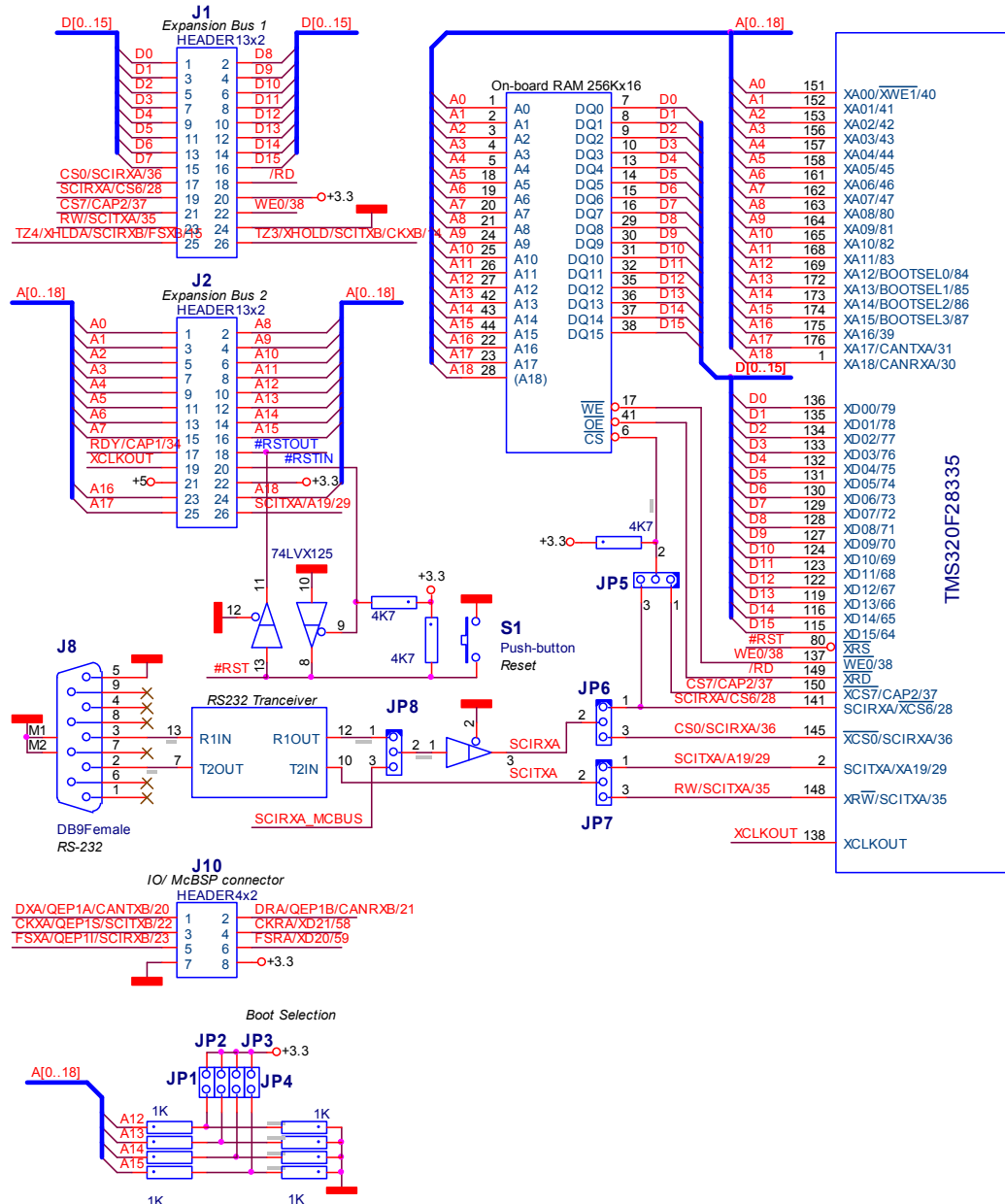


Figure A.3. Connections for J1, J2, J8, J10 signals

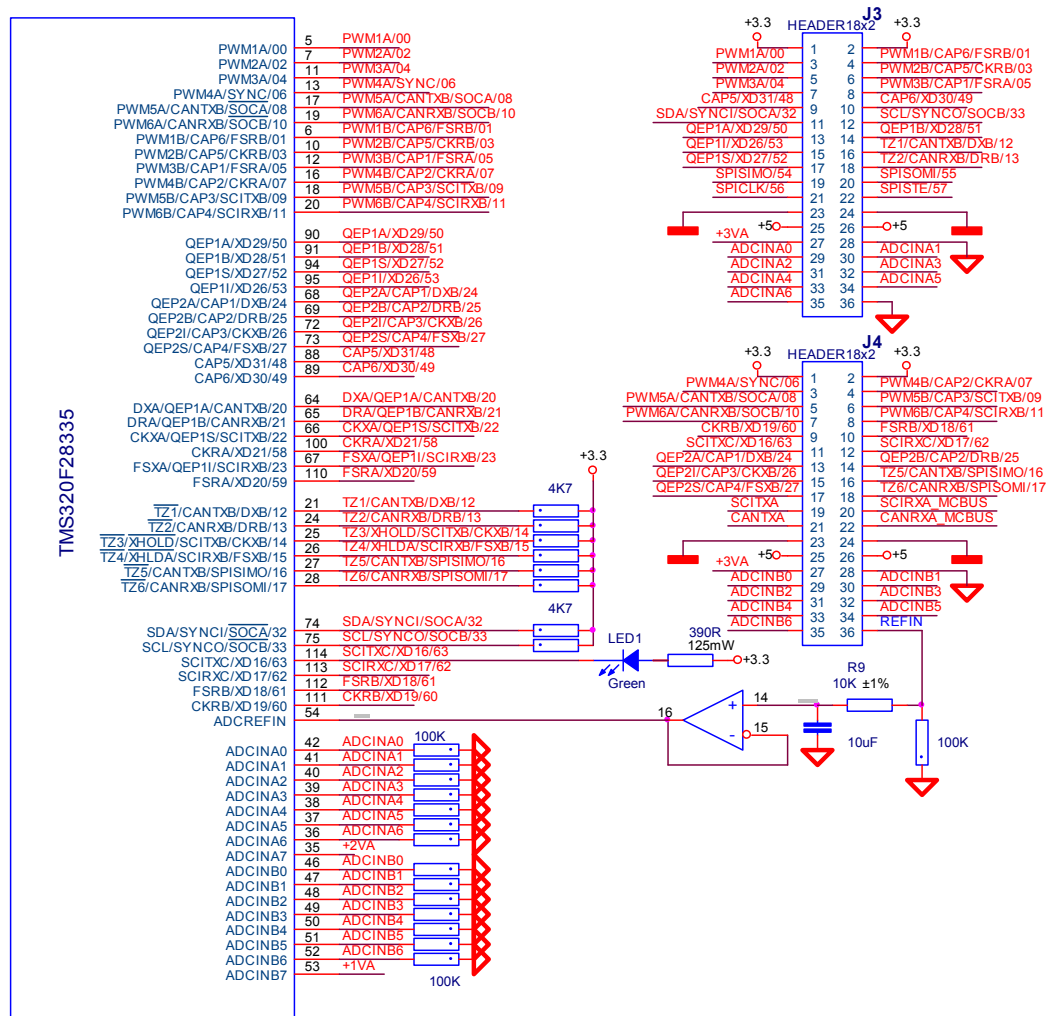


Figure A.4. Connections for J3 and J4 signals

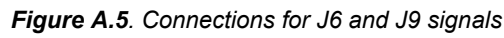


Figure A.6. MSK28335 – Jumper’s positions

Ref.	Type	Mounted default	Function																																																																																					
JP1, JP2, JP3, JP4	Jumper, 2 pins	Yes	<p>'F28335 operation after reset:</p> <table><thead><tr><th>JP4 GPIO87 / XA15</th><th>JP3 GPIO86 / XA14</th><th>JP2 GPIO86 / XA14</th><th>JP1 GPIO85 / XA13</th><th>BOOTMODE GPIO85 / XA13 GPIO84 / XA12</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>Jump to Flash (default)</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>SCI-A boot</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>SPI-A boot</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td><td>I2C-A boot</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>eCAN-A boot</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>McBSP-A boot</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>Jump to XINTF x16</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>Jump to XINTF x32</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>Jump to OTP</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>Parallel GPIO I/O boot</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>Parallel XINTF boot</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>Jump to SARAM</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>Branch to check boot mode</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>Branch to Flash, skip ADC calibration</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>Branch to SARAM, skip ADC calibration</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Branch to SCI, skip ADC calibration</td></tr></tbody></table> <p>1 = SHORT = jumper mounted 0 = OPEN = jumper not mounted</p> <p>Default JP1 = SHORT JP2 = SHORT JP3 = SHORT JP4 = SHORT</p>	JP4 GPIO87 / XA15	JP3 GPIO86 / XA14	JP2 GPIO86 / XA14	JP1 GPIO85 / XA13	BOOTMODE GPIO85 / XA13 GPIO84 / XA12	1	1	1	1	Jump to Flash (default)	1	1	1	0	SCI-A boot	1	1	0	1	SPI-A boot	1	1	0	0	I2C-A boot	1	0	1	1	eCAN-A boot	1	0	1	0	McBSP-A boot	1	0	0	1	Jump to XINTF x16	1	0	0	0	Jump to XINTF x32	0	1	1	1	Jump to OTP	0	1	1	0	Parallel GPIO I/O boot	0	1	0	1	Parallel XINTF boot	0	1	0	0	Jump to SARAM	0	0	1	1	Branch to check boot mode	0	0	1	0	Branch to Flash, skip ADC calibration	0	0	0	1	Branch to SARAM, skip ADC calibration	0	0	0	0	Branch to SCI, skip ADC calibration
JP4 GPIO87 / XA15	JP3 GPIO86 / XA14	JP2 GPIO86 / XA14	JP1 GPIO85 / XA13	BOOTMODE GPIO85 / XA13 GPIO84 / XA12																																																																																				
1	1	1	1	Jump to Flash (default)																																																																																				
1	1	1	0	SCI-A boot																																																																																				
1	1	0	1	SPI-A boot																																																																																				
1	1	0	0	I2C-A boot																																																																																				
1	0	1	1	eCAN-A boot																																																																																				
1	0	1	0	McBSP-A boot																																																																																				
1	0	0	1	Jump to XINTF x16																																																																																				
1	0	0	0	Jump to XINTF x32																																																																																				
0	1	1	1	Jump to OTP																																																																																				
0	1	1	0	Parallel GPIO I/O boot																																																																																				
0	1	0	1	Parallel XINTF boot																																																																																				
0	1	0	0	Jump to SARAM																																																																																				
0	0	1	1	Branch to check boot mode																																																																																				
0	0	1	0	Branch to Flash, skip ADC calibration																																																																																				
0	0	0	1	Branch to SARAM, skip ADC calibration																																																																																				
0	0	0	0	Branch to SCI, skip ADC calibration																																																																																				
JP5, JP6, JP8	Jumper, 3 pins	Yes	<p>SCIRX connection:</p> <p>JP8 JP6 JP5 Signal connection 1-2 1-2 1-2 GPIO37 / ECAP2 / XZCS7 is connected to CS (chip select) pin of External RAM GPIO28 / SCIRXDA / XZCS6 is connected to RS232 transceiver output; GPIO36 / SCIRXDA / XZCS0 not connected 1-2 1-2 2-3 NOT ALLOWED! THE DSC CAN BE DAMAGED! GPIO37 / ECAP2 / XZCS7 not connected GPIO28 / SCIRXDA / XZCS6 is connected to CS (chip select) pin of External RAM and RS232 transceiver output; CONFLICT ! GPIO36 / SCIRXDA / XZCS0 not connected 1-2 2-3 1-2 GPIO37 / ECAP2 / XZCS7 is connected to CS (chip select) pin of External RAM GPIO28 / SCIRXDA / XZCS6 not connected; GPIO36 / SCIRXDA / XZCS0 is connected to RS232 transceiver output 1-2 2-3 2-3 GPIO37 / ECAP2 / XZCS7 is not connected GPIO28 / SCIRXDA / XZCS6 not connected; GPIO36 / SCIRXDA / XZCS0 is connected to RS232 transceiver output 2-3 1-2 1-2 GPIO37 / ECAP2 / XZCS7 is connected to CS (chip select) pin of External RAM GPIO28 / SCIRXDA / XZCS6 is connected to SCIRXA_MCBUS; GPIO36 / SCIRXDA / XZCS0 not connected 2-3 1-2 2-3 NOT ALLOWED! THE DSC CAN BE DAMAGED!</p>																																																																																					

			<p>GPIO37 / ECAP2 / XZCS7 is not connected GPIO28 / SCIRXDA / XZCS6 is connected to CS (chip select) pin of External RAM and SCIRXA_MCBUS; GPIO36 / SCIRXDA / XZCS0 not connected 2-3 2-3 1-2 GPIO37 / ECAP2 / XZCS7 is connected to CS (chip select) pin of External RAM GPIO28 / SCIRXDA / XZCS6 not connected GPIO36 / SCIRXDA / XZCS0 is connected to SCIRXA_MCBUS; 2-3 2-3 2-3 GPIO37 / ECAP2 / XZCS7 is not connected; GPIO28 / SCIRXDA / XZCS6 not connected; GPIO36 / SCIRXDA / XZCS0 is connected to SCIRXA_MCBUS;</p> <p>Default JP5 = 1-2 JP6 = 1-2 JP8 = 1-2</p>
JP7	Jumper, 3 pins	Yes	<p>TXD pin (J4/19) connection:</p> <p>1-2 = TXD is connected to GPIO29 / SCITXDA / XA19 pin of 'F28335 and TX of RS232 transceiver 2-3 = TXD is connected to GPIO35 / SCITXDA / XR / W pin of 'F28335</p> <p>Default = 1-2</p>
JP9	Jumper, 3 pins	Yes	<p>GPIO18 / SPICLKA / SCITXDB / CANRXA pin connection:</p> <p>1-2 = GPIO18 / SPICLKA / SCITXDB / CANRXA pin of 'F28335 is connected to CANRx of CAN transceiver 2-3 = CANRXA is connected to CANRXA_MCBUS pin of 'F28335</p> <p>Default = 1-2</p>
JP10	Jumper, 2 pins	Yes	<p>CAN-bus terminating resistor:</p> <p>SHORT = CAN-bus 120Ω terminating resistor is connected between CAN-HI and CAN-LO OPEN = CAN-bus 120Ω terminating resistor not connected</p> <p>Default = SHORT</p>
JP11	Jumper, 2 pins	Yes	<p>GPIO57 / SPISTEA / XD22 pin connection:</p> <p>SHORT = GPIO57 / SPISTEA / XD22 pin of 'F28335 is connected to CS (chip select) of external EEPROM memory OPEN = GPIO57 / SPISTEA / XD22 pin is not connected to CS (chip select) of external EEPROM memory</p> <p>Default = SHORT</p>

A.4. MSK28335 DSC Board - Electrical Specifications

Recommended Operating Conditions

PARAMETER [†]		TEST CONDITIONS	Min	Typ	Max	UNIT
DC Power Input						
V _{DD}	Supply voltage		4.75	5	5.25	V
I _{DD}	Supply current	V _{DD} =5V, I _O =0 all outputs			350	mA
Digital Inputs (J3, J4, J10 headers)						
V _{IH}	High-level input voltage	All inputs	2		3.3+0.3	V
V _{IL}	Low-level input voltage	All inputs	−0.3		0.8	
Analog Inputs (J3, J4, J10 headers)						
V _{IA}	Analog inputs range	V _{DD} = 3.3V	−0.3	0 ...3.0	3.0	V
I _{IA}	Analog input source impedance	V _{IA} = GND to 3.0V			100	Ω
Digital Outputs (J3, J4, J10 headers)						
V _{OH}	High-level output voltage		2.4			V
V _{OL}	Low-level output voltage				0.6	V
System Environment						
T _A	Operating ambient temperature		0		50	°C

A.5. MSK28335 DSC Board - Mechanical drawing

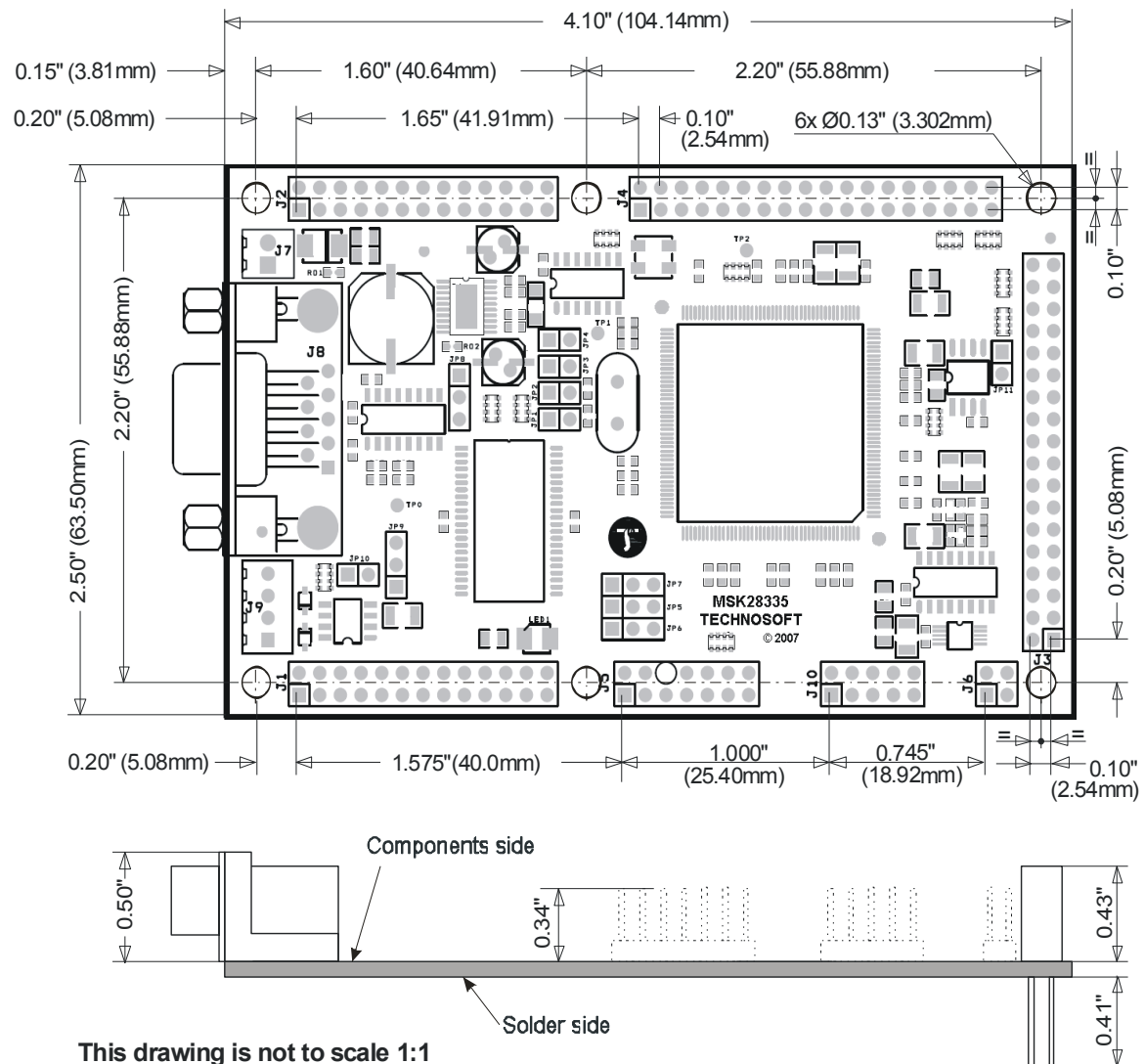


Figure A.6. MSK28335 – Mechanical drawing

Appendix B. Connecting external power modules to MSK28335 DSC board

The MSK28335 DSC board can be connected with one or 2 external power modules, using the two headers J3 and J4 through which almost all 'F28335 I/O signals are accessible. Some of the I/O signals are also used on MSK28335 board. However, with minor changes presented below (see also Appendix A) these signals also become available on headers.

This appendix presents the settings and the rules to follow when MSK28335 DSC board is connected with an external power module. All the solder-joints referred below are located on the topside of the board. All of them are clearly marked.

**CAUTION!**

READ CAREFULLY ALL THE FOLLOWING INSTRUCTIONS! IT IS THE USER RESPONSIBILITY IF BOARD MALFUNCTIONING OR DAMAGE IS CAUSED BY AN IMPROPER CONFIGURATION OF THE MSK28335 BOARD.

1. Be sure that:

- **All digital signals coming from the power module are 3.3V compatible**
- **All analog signals coming from the power module are 3V compatible.**
- **If the power module outputs are 0-5V, a 5-to-3.3V level translator is required.**

- 2.** Disconnect the MSK28335 external power supply applied on J7 connector. The MSK28335 will be powered exclusively through the J3 and J4 extension connectors, with **+4.75...+5.25V_{DC}** applied on ALL the following pins: **25, 26 / J3 and 23, 24 / J4** (named **+5V_{DC}**). The supply return (ground) must go on pins **23, 24 / J3 and 23, 24 / J4** (named **GND**). A minimum of 310 mA is required for MSK28335 DSC board current supply. Note that the supply return **GND** is galvanic connected on the MSK28335 DSC board with the analog ground pins of the MC-BUS connectors (**VREFLO**). To avoid ground loops use **GND** only for supply and as reference for the digital signals, while **VREFLO** only as reference for the analog signals. A suggested grounding scheme is shown in figures **B.1 / B.2**.

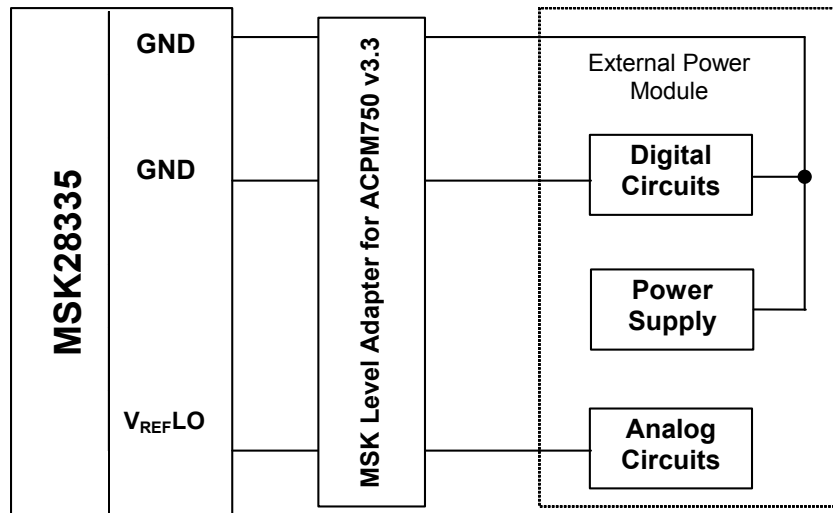


Figure B.1 Recommended ground connections – ACPM750 v3.3

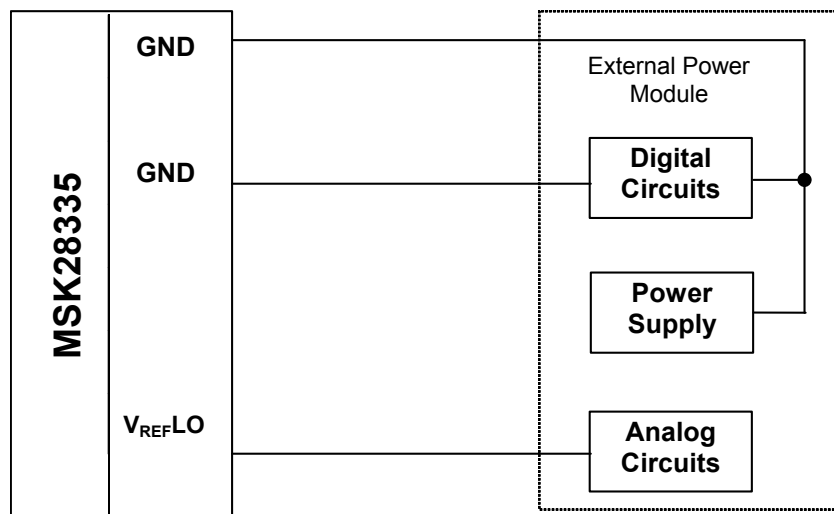


Figure B.2 Recommended ground connections – ACPM750 v3.4

3. By default the MSK28335 uses its DB9 connector for serial communication and **GPIO28 / SCIRXDA / XZCS6** and **GPIO29 / SCITXDA / XA19** signals are connected to on-board RS-232 transceiver. In order to send/receive serial (SCI) data using the power module transceiver, or any other device connected to J4, you need to configure JP8 on 2-3 position or to use another SCI-A pins (**GPIO36 / SCIRXDA / XZCS0** and **GPIO35 / SCITXDA / XR / W**).

Both signals on J4 (RXD and TXD) are 3.3V compatible.

4. The MSK28335 DSC board **SPISIMO / SPISOMI** and **SPICLK** signals are connected to 2 on-board devices: the E2ROM and the DAC. In order to send/receive data through SPI from an external device connected to **J3**, you need to disable one of these devices and enable the external device by configuring JP10:
 - On board E2ROM and external SPI device: **JP11 = SHORT (SPISTE = LOW selects the E2ROM)**
 - Only external SPI device: **JP11 = SHORT (SPISTE = LOW selects the E2ROM)**
 - On board DAC device:
 - **JP11 = SHORT (SPISTE = HIGH and GPIO61 = LOW selects the DAC)**
 - **JP11 = OPEN (SPISTE = X and GPIO61 = LOW selects the DAC)**
5. By default the MSK28335 uses its on-board transceiver for CAN communication and **GPIO19 / SPISTE_A / SCIRXDB / CANTXA** and **GPIO18/ CANRXA/ SPICLK/ SCITXB** signals are connected to it. In order to send/receive serial CAN-Bus data using the power module transceiver, or any other device connected to J4, you need to configure JP9 on 2-3 position or to use another CAN-A pins (**GPIO30 / CANRXA / XA18** and **GPIO31 / CANTXA / XA17**).

Both signals on J4 (CANRXA and CANTXA) are 3.3V compatible.
6. Keep in mind the following fan-out and fan-in restrictions:
 - **GPIO63/ SCITXC/ XD16** 'F28335 signal drives the on-board green led. If this signal is used as inputs, the I_{OL} requirement is minimum 4mA.
 - **You can't use GPIO28 / SCIRXDA / XZCS6 and GPIO18/ CANRXA/ SPICLK/ SCITXB as digital output.**
 - **VREFHI** analog reference output signal is the +3.0V_{DC}, separated by LC filter. Do not make short-circuits to GND, +5V or other potentials. Do not introduce any spurious noise on this signal, otherwise the A/D conversion performance will be affected. Note that the nominal voltage of this signal has an absolute accuracy of $\pm 0.1V$.

The next picture presents the connection of the MSK28335 board to the PM50 power module.

Plug-in J3 of MSK28335 to J3 of PM50 v3.x

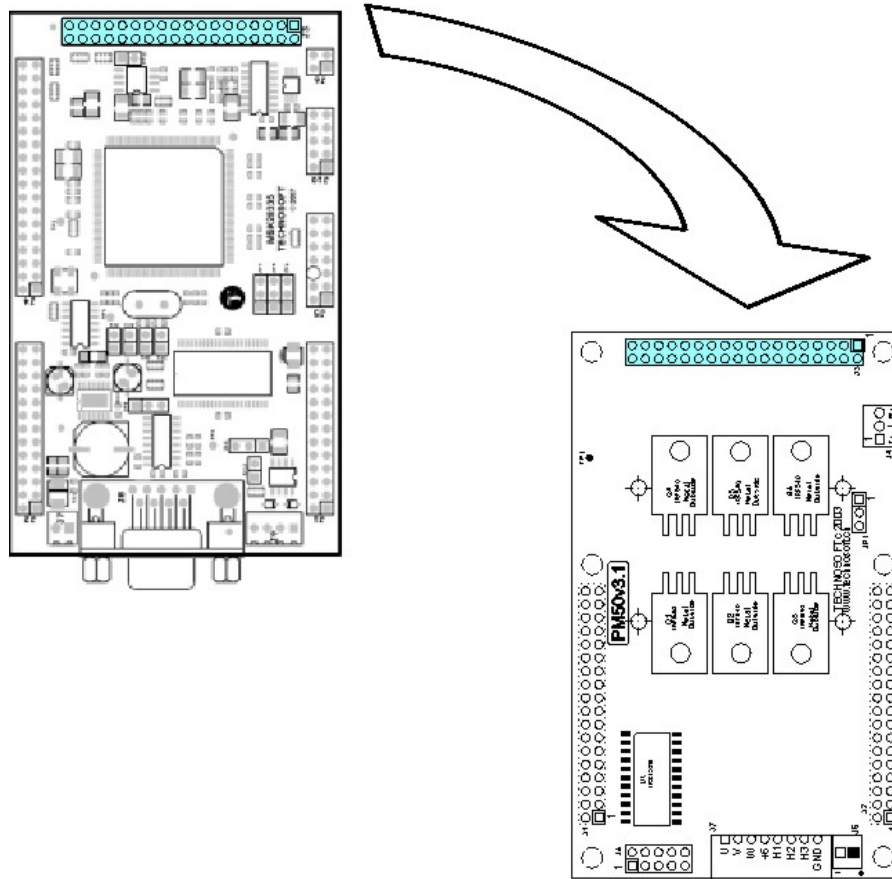


Figure B.2 Mounting MSK28335 board on PM50

The next picture presents the connection of the MSK28335 board to the ACPM750 power module through MSK Level Adapter board.

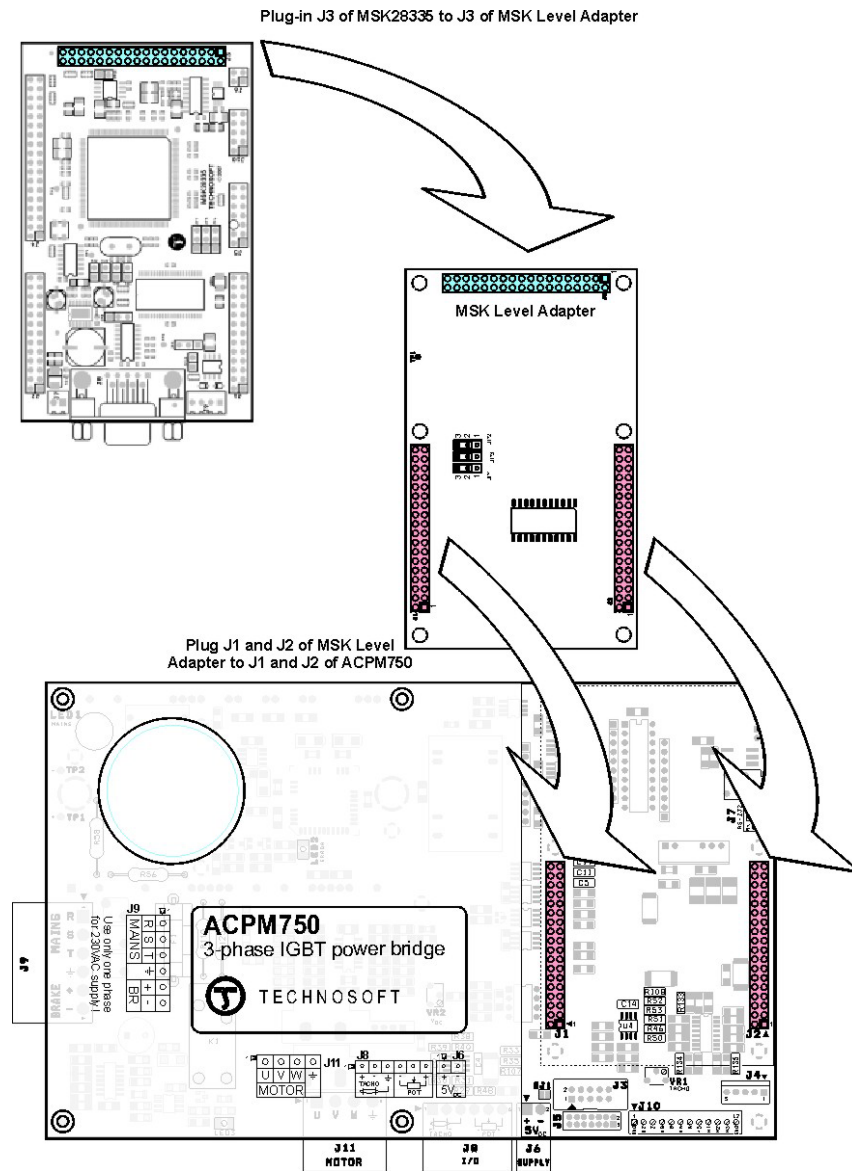


Figure B.3 Mounting MSK28335 board on ACPM750 v3.3

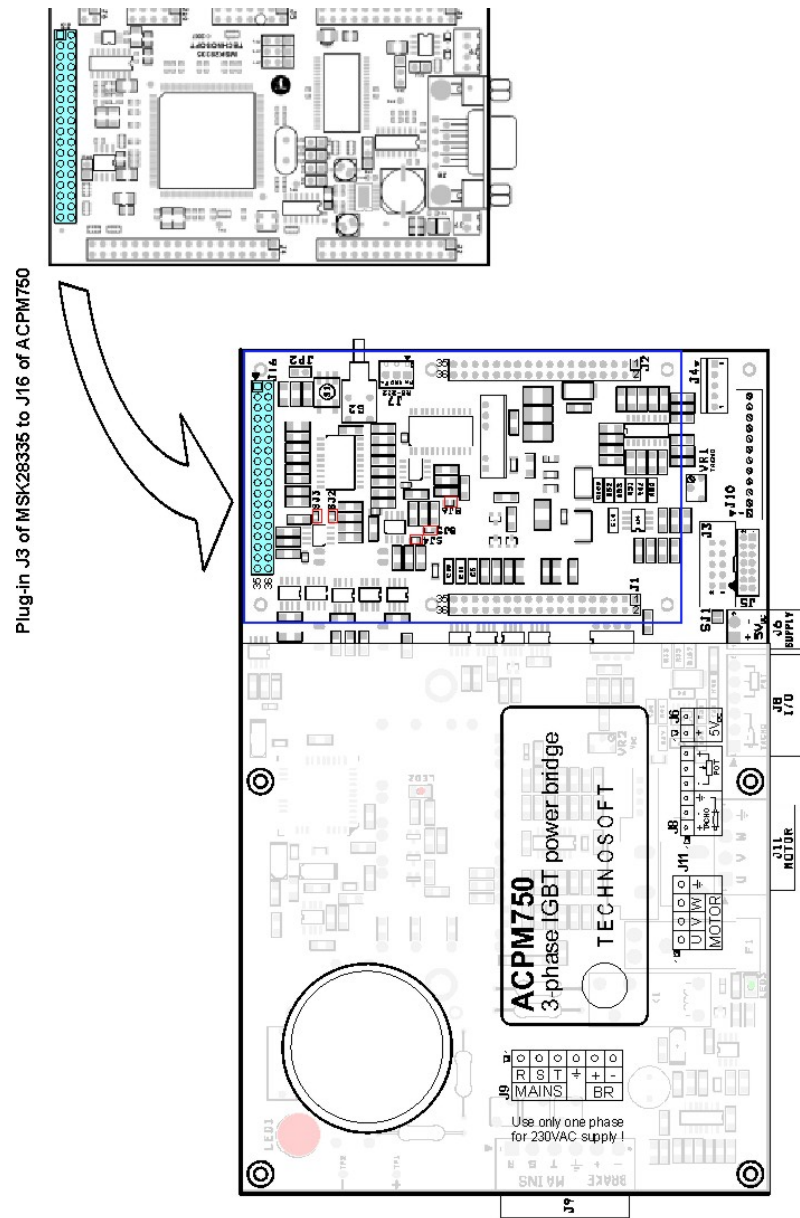


Figure B.4 Mounting MSK28335 board on ACPM750 v3.4

Appendix C. TMS320F28335 resources used by MSK28335 board

C.1. Memory resources used by MON28335 monitor

Memory type and range	Used by MON28335	Program Type	Observations
Flash EEPROM PS,DS: 0x33FFF6÷0x33FFF7	0x33FFF6÷0x33FFF7	Branch to monitor code	Reserved
Flash EEPROM PS,DS: 0x330000÷0x337FFF	0x330000÷0x337FFF	Program memory used by MON28335 monitor	Reserved
On-chip DRAM L0 (dual-mapped) PS, DS: 0x3F8000÷0x3F83FF PS, DS: 0x8000÷0x83FF	0x3F8000÷0x3F83FF 0x8000÷0x83FF	Data memory used by MON28335 monitor	Reserved
On-chip RAM M0 PS, DS: 0x0000÷0x03FF	0x0000÷0x03FF	MON28335 SP starting address	SP can freely be reassigned. Space used by SP depends on program complexity

Legend: PS = program memory space, DS = data memory space, SP = software stack pointer

C.2. TMS320F28335 I/O signals used on MSK28335 DSC board

RS-232 serial communication

GPIO29/ SCITXA/ A19	Sends data to on-board RS-232 transceiver
GPIO28/ SCIRXA/ CS6	Gets data from on-board RS-232 transceiver

CAN communication

GPIO19/ CANTXA/ SPISTE/ SCIRXB	Sends data to on-board CAN Bus transceiver
GPIO18/ CANRXA/ SPICLK/ SCITXB	Reads data from on-board CAN Bus transceiver

LED drivers

GPIO63/ SCITXC/ XD16	Drives the on-board green LED
----------------------	-------------------------------

SPI communication

GPIO54/ SPISIMO GPIO55/ SPISOMI GPIO56/ SPICLK GPIO57 / SPISTE	Used by on-board EEPROM or by other external devices (Resolver-to-digital interface RDIM16, ACPM750E, etc.
GPIO54/ SPISIMO GPIO56/ SPICLK GPIO61/ FSRB / XD18 GPIO60/ CKRB / XD19	Used by on-board DAC

Other used signals

GPIO13/ TZ2 / CANRXB / DRB	Used by ACPM750 external power module (if connected)
All J3 signals (14 signals)	If the MSK28335 DSC board is connected with the PM50 or ACPM750E power module

Appendix D. PM50 V3.1 power module technical data

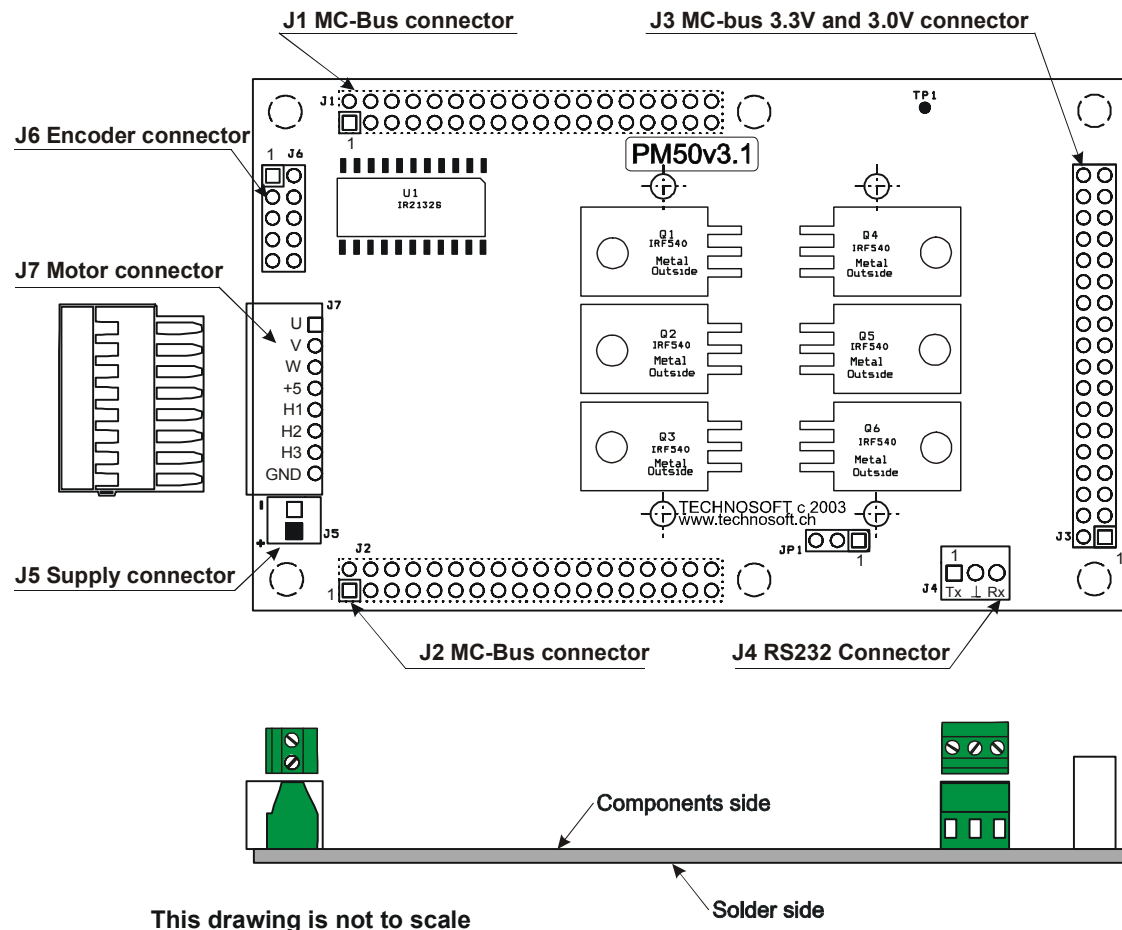
Pm50 V3.1 was designed to work with both 5V and 3.3V I/O signal levels.

It may function with MSK240 / 243 / 2407 / 28335 boards using J1, J2 connectors (5V MC bus).

It may function with MSK2407 (without using MSK Level Adapter board) using **only** J3 connector (3.3V bus - analog & digital).

It may function with MSK28335 (without using MSK Level Adapter board) using **only** J3 connector (3.0V bus for analog signals and 3.3V bus for digital signals).

D.1. PM50 V3.1 - Connectors



Note:

1. The Wago 90° female connector for the motor (the external one) is supplied with the motor.
2. Connect the power supply (on J5 connector) before plug-in the MSK28335 DSC board into PM50 V3.1.
3. Always MSK28335 DSC board should be on top of PM50 V3.1.

D.2. Power Supply

J5 – Supply Connector

The PM50 V3.1 requires only one power supply: 9 – 36 V_{DC} for the motor. Other required voltages including 5V_{DC} for the internal logic are provided by on board power supplies directly from motor power supply. The motor power supply is applied on the inverter DC-bus and must be adapted to the motor and application requirements. It should be able to absorb the energy generated during motor braking.

CAUTION !

Be careful to correctly connect the power supplies polarities! Reverse polarities will damage the PM50 module!

The PM50 V3.1 is equipped with a 2-pin 0.1" pitch screw-terminal connector for the supply. Its pin assignment is presented in Table D.1.

Table D.1. J5 – Supply Connector: 2-pin 0.1" pitch screw-terminal connector

Pin	Name	Type	Function
1	+MOT	I	9 - 36V power supply plus (+)
2	GND	I	9 - 36V power supply minus (-)

D.3. Power Stage Module

The PM50 V3.1 power module includes a 3-phase inverter, the protection circuits and the measurement circuits for the DC-bus voltage and the motor currents.

3-phase Inverter Command

The 3-phase inverter (see figure E.1) uses MOSFET transistors with switching frequency up to 50kHz. The PM50 V3.1 MC-bus interface includes 6 PWM command inputs (TTL/CMOS compatible) named $\overline{\text{PWM1A}}$, $\overline{\text{PWM1B}}$, $\overline{\text{PWM2A}}$, $\overline{\text{PWM2B}}$, $\overline{\text{PWM3A}}$, $\overline{\text{PWM3B}}$ through which the control unit can drive each transistor of the inverter.

All PWM commands are active low.

- $\overline{\text{PWM1A}}$, $\overline{\text{PWM2A}}$, $\overline{\text{PWM3A}}$ drive the upper transistors.
- $\overline{\text{PWM1B}}$, $\overline{\text{PWM2B}}$, $\overline{\text{PWM3B}}$ drive the lower transistors.
- $\overline{\text{PWM1A}}$ & $\overline{\text{PWM1B}}$ drive phase A,
- $\overline{\text{PWM2A}}$ & $\overline{\text{PWM2B}}$ drive phase B,
- $\overline{\text{PWM3A}}$ & $\overline{\text{PWM3B}}$ drive phase C.

The control unit PWM commands should include a dead time of minimum 0.5μs for transistor's commutation.

Voltage input levels for PWM commands are 5V on J1 connector and 3.3V on J3 connector - direct compatible with MSK28335.

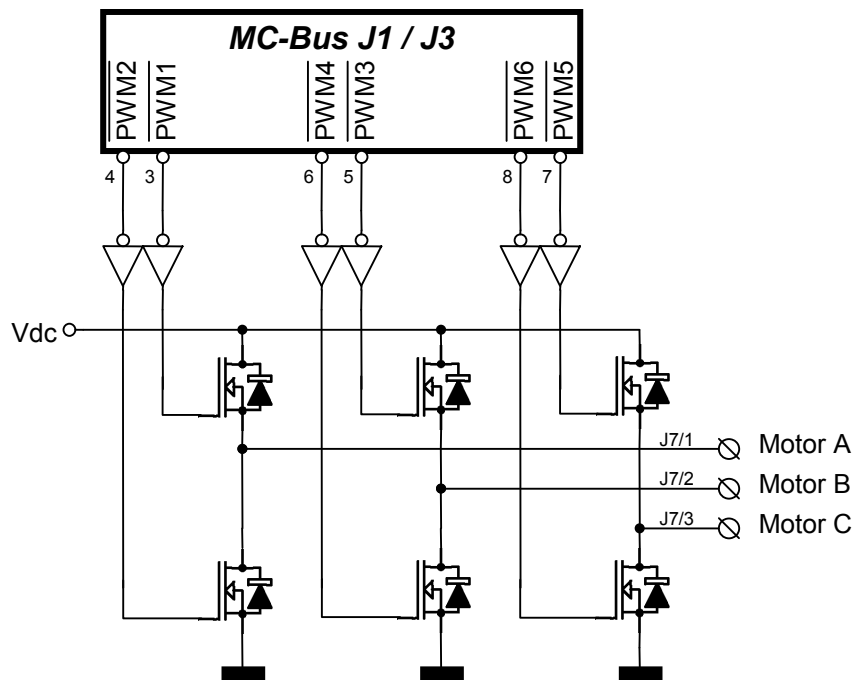


Figure D.1. 3-phase Inverter Control using PWM Command Inputs

Protections. Error Signal

The PM50 V3.1 power stage includes protection circuits for: phase to phase short-circuit, over-current, over-voltage and accidental shutdown of logic supply (logic supply under-voltage). If any of these protections is triggered, the PM50 V3.1 sets low the MC-bus error signal $\overline{\text{PDPINT}}$ (open-collector signal) to sign this event to the control unit and the inverter transistors are turned off as long as the condition is maintained. If any low-leg motor current exceeds 7A for more than 10 ms, a $\overline{\text{PDPINT}}$ fault signal will occur. Short-term over-voltages having energy of less than 3J ($47\text{V} \times 5\text{A} \times 12\text{ms}$) will be clamped by an on-board 47V suppresser.

Current Measurement Interface

Using J1/J2 MC bus (5V)

The PM50 v3.1 measures the motor currents using shunts placed in the lower-legs of the inverter (see Figure D.2) using a current gain factor of 0.395V/A, which translates $\pm 6.33\text{A}$ currents into $\pm 2.5\text{V}$ voltages. These voltages are applied on the 5V J2 MC-bus analogue inputs IA, IB and IC with a 2.5V offset. Hence negative currents are measured between 2.5V to 5V and positive currents are measured between 0 and 2.5V. The midpoint 2.5V corresponds to a zero current.

$$\text{Phase current [A]} = (2.5\text{V offset} - \text{voltage on analogue input [V]}) / \text{current gain [V/A]}$$

Using J3 MC bus (3.0V or 3.3V)

The same current signals are translated from 0-5V level range in 0-3.0V or 0-3.3V level range using analogue interface for J3 connector. In that case, for 0 – 3.0V range, the midpoint 1.5V corresponds to a zero current and the current gain factor is $0.395\text{V/A} \cdot 2 / 3.3 = 0.239\text{V/A}$, and for the 0 – 3.3V range the midpoint 1.5V corresponds to a zero current and the current gain factor is $0.395\text{V/A} \cdot 2 / 3 = 0.263\text{V/A}$.

Remark: A negative current is defined as flowing from the lower transistor/diode to ground e.g. when the current exits from the motor. A positive current is defined as flowing from ground to the lower transistor/diode e.g. when the current enters in the motor. The motor phase currents can be computed with the formula:

$$\text{Phase current [A]} = (\text{offset (1.5V or 1.65V)} - \text{voltage on analogue input [V]}) / \text{current gain [V/A]}$$

WARNING: Be aware that the current measurement scheme, simple and cost-effective from the hardware point of view, requires some special care from the point of view of software implementation. Thus, for the three currents measurement scheme, the A/D conversion **MUST** be synchronised with the PWM command of the inverter transistors, to properly measure the currents on each phase of the motor.

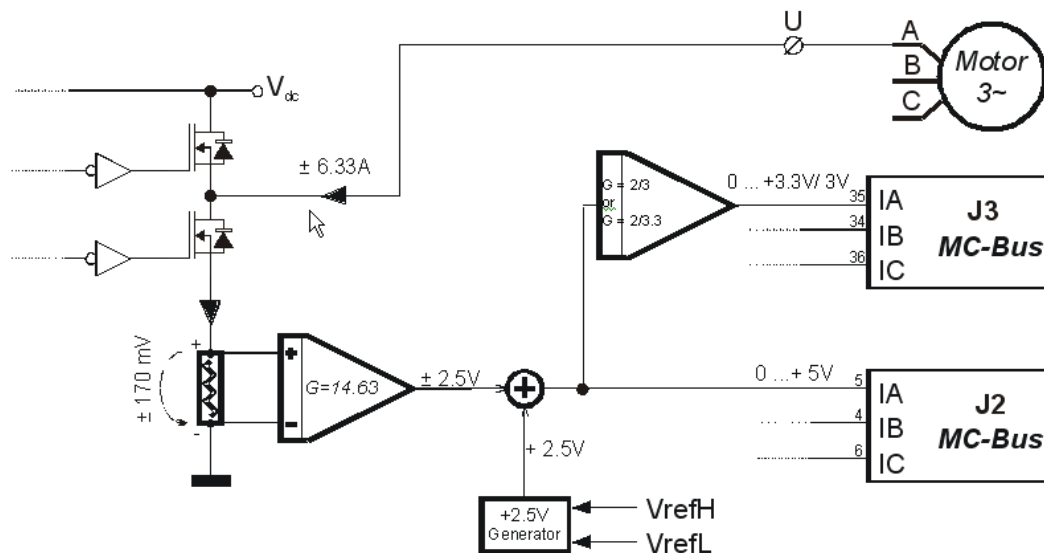


Figure D.2. Current Measurement

Motor Supply / DC-bus Voltage Measurement

The PM-50 v 3.1 includes a motor supply / DC-bus voltage feedback applied on the MC-bus analogue input named VDC. The gain factor is 0.0909. Hence, DC-bus voltages between 0 and 55V are translated into 0 to 5V at J2 connector / pin 3.

For 3.0V bus (J3), the gain factor is $0.0909 \cdot 2 / 3.3 = 0.055$, and for 3.3V bus (J3) the gain factor is $0.0909 \cdot 2 / 3 = 0.0606$. Hence, DC-bus voltages between 0 and 55V are translated into 0 to 3.0V at J3 connector / pin 33.

Please note that the 55V is only a computing value! Do not supply the board with more than 36V if your power supply can not absorb the energy produced during the braking stage of the motor (especially when the load has a high inertia). The PM50 V3.1 board has an internal protection (a Zener Diode of 47V, 1W)

The measurement scheme provides filtering with a cut-off frequency of 730 Hz in order to remove spurious switching noise generated by the on-board power stage.

D.4. Motor sensors

J7 – Motor and Hall Sensors Connector

The PM50 v3.1 is equipped with an 8-pin 0.1" pitch Wago 90° male connector for the motor. Its pin assignment is presented in Table D.2.

Table D.2. J7 – Motor Connector: 8-pin 0.1" pitch Wago 90° male connector

Pin	Name	Type	Function
1	U	O	Phase A for 3-phase motors
2	V	O	Phase B for 3-phase motors
3	W	O	Phase C for 3-phase motors
4	+5V	O	+5V _{DC} for Hall sensors
5	HALL1	I	Hall 1
6	HALL2	I	Hall 2
7	HALL3	I	Hall 3
8	GND		Ground

The output level for Hall sensors is 0-5V – on the J2 connector, or 0-3.3V on J3 connector.

J6 – Encoder Connector

By default, the PM50 v3.1 is equipped with a 2x5 0.1" pitch header connector for the encoder. Its pin assignment is presented in Table D.3. The encoder connector can be configured from **SJ1** solder joint (placed on the solder side near J6 connector) to be compatible with API-Portescap A22 encoders or Hewlett Packard/Agilent HEDL65xx encoders. The **SJ1** solder joint must be:

- **ON** for HP HEDL65xx encoder type
- **OFF** for API-Portescap A22 encoder type (default)

Table D.3. J6 – Encoder Connector (default): 2x5 0.1" pitch header connector

Pin	Name	Type	Function
1	GND		Ground
2	+5V	O	Encoder supply
3	DIR		Connected to ground if SJ1 = ON (see note)
4	STBY	I	Connected to +5V
5	-	n.c.	Not connected
6	A	I	Positive A for differential encoder or A for single-ended encoder
7	-	n.c.	Not connected
8	B	I	Positive B for differential encoder or B for single-ended encoder
9	-	n.c.	Not connected
10	Z	I	Positive Z for differential encoder or Z for single-ended encoder

On request, the PM50 v3.1 can also be equipped with a vertical (J666) or 90° (J66) 5-pin Molex connector for the encoder, which is compatible with all Hewlett Packard/Agilent HEDSxxxx single-ended TTL-compatible encoders. Table D.4 presents the pin assignment the for Molex connectors.

Table D.4. J66 – Encoder Connector (alternate option): 5-pin Molex connector

Pin	Name	Type	Function
1	GND		Ground
2	Z	I	Positive Z for differential encoder or Z for single-ended encoder
3	A	I	Positive A for differential encoder or A for single-ended encoder
4	+5V	O	Encoder sensor supply
5	B	I	Positive B for differential encoder or B for single-ended encoder

D.5. Communication

J4 – RS-232 Connector

The PM50 v3.1 power module includes an RS-232 connector. It can be used to pass the Rx232 and Tx232 signals to the control unit through the MC-bus interface. The PM50 v3.1 simply applies the signals from the RS-232 connector to the J2 MC-bus. The RS-232 transceiver should be on the control unit.

The PM 50v3.1 is equipped with a 3-pin 0.1" pitch screw-terminal connector for RS-232 link. Its pin assignment is presented in Table D.5.

Table D.5. J4 – RS-232 Connector (default): 3-pin 0.1" pitch screw-terminal connector

Pin	Name	Type	Function
1	TxD	O	Data Transmission
2	GND		Ground
3	RxD	I	Data Reception

Note: Do not use this connector for serial communication when PM50 is used together with the MSK28335 DSC board. Use **only** the DB9 serial connector J8 from the MSK28335 DSC board.

D.6. Control Unit Interface (MC-Bus)

J1, J2, J3 – MC-Bus Connectors

The PM50 v3.1 power module receives the PWM command signals and sends the feedback and status signals to the control unit through two 2x18-pin 0.1" pitch MC-Bus connectors J1 and J2 for 5V levels or through J3 – for 3.3V levels.

Table D.6. J1 – MC-bus connector

Pin	Name	Type	Description
1	-reserved-	n.c.	Not connected on the PM50 v3.1
2	-reserved-	n.c.	
3	PWM1	I	CU output. Active low command for the upper transistor, A-phase leg
4	PWM2	I	CU output. Active low command for the lower transistor, A-phase leg
5	PWM3	I	CU output. Active low command for the upper transistor, B-phase leg
6	PWM4	I	CU output. Active low command for the lower transistor, B-phase leg
7	PWM5	I	CU output. Active low command for the upper transistor, C-phase leg
8	PWM6	I	CU output. Active low command for the lower transistor, C-phase leg
9	-reserved-	n.c.	Not connected on the PM50 v3.1
10	-reserved-	n.c.	Not connected on the PM50 v3.1
11	-reserved-	n.c.	Not connected on the PM50 v3.1
12	-reserved-	n.c.	Not connected on the PM50 v3.1
13	-reserved-	n.c.	Not connected on the PM50 v3.1
14	-reserved-	n.c.	Not connected on the PM50 v3.1
15	-reserved-	n.c.	Not connected on the PM50 v3.1
16	-reserved-	n.c.	Not connected on the PM50 v3.1
17	ENC-A	O	CU input. Encoder signal A from J6, J66 or J666
18	ENC-B	O	CU input. Encoder signal B from J6, J66 or J666
19	ENC-Z	O	CU input. Encoder signal Z (index/zero) from J6, J66 or J666
20	-reserved-	n.c.	Not connected on the PM50 v3.1
21	PDPINT	O	CU input. Power stage error signal, driven by protection circuitry. Open-collector signal
22	-reserved-	n.c.	Not connected on the PM50 v3.1
23	-reserved-	n.c.	
24	-reserved-	n.c.	
25	+5V _{DC}	O	+5V _{DC} output
26	DGND		Ground terminal. All digital signals are referenced to this signal
27	DGND		
28	DGND		
29	DGND		
30	-reserved-	n.c.	Not connected on the PM50 v3.1
31	-reserved-	n.c.	
32	-reserved-	n.c.	
33	-reserved-	n.c.	
34	-reserved-	n.c.	
35	-reserved-	n.c.	
36	-reserved-	n.c.	

Table D.7. J2 – MC-bus connector

Pin	Name	Type	Description
1	-reserved-	n.c.	Not connected on the PM50 v3.1
2	-reserved-	n.c.	
3	VDC	O	CU analog input. Value of the motor supply / DC-bus voltage. Gain factor is 91mV/V
4	IB	O	CU analog input. Value of current in B-phase inverter lower leg, upper shifted with 2.5V. Gain factor is 0.395V/A
5	IA	O	CU analog input. Value of current in A-phase inverter lower leg, upper shifted with 2.5V. Gain factor is 0.395V/A
6	IC	O	CU analog input. Value of current in C-phase inverter lower leg, upper shifted with 2.5V. Gain factor is 0.395V/A
7	V _{REFLO}	O	Analogue ground. Separated from DGND. The connection between these two signals must be done on the CU
8	+5V _A	I	Analogue reference. CU must provide the +5V _A . Half of this value represents the upper shift for bipolar signals. Typical expected value is +5V, allowed range is +4.75...5.25V. Current drawn by PM50 v3.1 is under 0.5 mA
9	DGND		Ground terminal. All digital signals are referenced to this signal
10	DGND		
11	Hall 1	O	CU input. HALL1 signal from J7 connector. TTL signal (0-5V)
12	-reserved-	n.c.	Not connected on the PM50 v3.1
13	Hall 2	O	CU input. HALL2 signal from J7 connector. TTL signal (0-5V)
14	Hall 3	O	CU input. HALL3 signal from J7 connector. TTL signal (0-5V)
15	DGND		Ground terminal. All digital signals are referenced to this signal
16	-reserved-	n.c.	Not connected on the PM50 v3.1
17	Rx232	O	CU RS-232 input linked directly to J4 connector pin 3
18	Tx232	I	CU RS-232 output, linked directly to J4 connector pin 1
19	-reserved-	n.c.	Not connected on the PM50 v3.1
20	-reserved-	n.c.	
21	+5V _{DC}	O	+5V _{DC} output
22	+5V _{DC}	O	
23	-reserved-	n.c.	Not connected on the PM50 v3.1
24	-reserved-	n.c.	Not connected on the PM50 v3.1
25	-reserved-	n.c.	Not connected on the PM50 v3.1
26	-reserved-	n.c.	Not connected on the PM50 v3.1
27	-reserved-	n.c.	Not connected on the PM50 v3.1
28	-reserved-	n.c.	Not connected on the PM50 v3.1
29	-reserved-	n.c.	Not connected on the PM50 v3.1
30	-reserved-	n.c.	Not connected on the PM50 v3.1
31	In	O	CU input. Provides JP1 position: 1 logic -> JP1 =1-2, 0 logic -> JP1=2-3
32	-reserved-	n.c.	Not connected on the PM50 v3.1
33	-reserved-	n.c.	Not connected on the PM50 v3.1
34	-reserved-	n.c.	Not connected on the PM50 v3.1
35	-reserved-	n.c.	Not connected on the PM50 v3.1
36	-reserved-	n.c.	Not connected on the PM50 v3.1

Table D.8. J3 – connector (3.3V-bus for digital signals and 3.0V-bus for analog signals)

Pin	Name	Type	Description
1	+3.3V	I	+3.3V Power supply input
2	+3.3V	I	
3	PWM1A	I ¹	CU ² output. Active low command for the upper transistor, A-phase leg
4	PWM1B	I	CU output. Active low command for the lower transistor, A-phase leg
5	PWM2A	I	CU output. Active low command for the upper transistor, B-phase leg
6	PWM2B	I	CU output. Active low command for the lower transistor, B-phase leg
7	PWM3A	I	CU output. Active low command for the upper transistor, C-phase leg
8	PWM3B	I	CU output. Active low command for the lower transistor, C-phase leg
9	-	n.c.	Not connected on the PM50 v3.1
10	GPIO49 / ECAP6 / XD30	O ³	3.3V level translated output Hall1 from J7 connector.
11	GPIO32 / SDAA / EPWMSYNCI / ADCSOCAO	O	3.3V level translated output Hall2 from J7 connector.
12	GPIO33 / SCLA / EPWMSYNCO / ADCSOCBO	O	3.3V level translated output Hall3 from J7 connector.
13	GPIO50 / EQEP1A / XD29	O	Output QEP A. Encoder signal A from J6, J66 or J666. 3.3V level translated
14	GPIO51 / EQEP1B / XD28	O	Output QEP B. Encoder signal B from J6, J66 or J666. 3.3V level translated
15	GPIO53 / EQEP1I / XD26	O	Output QEP Z. Encoder signal Z from J6, J66 or J666. 3.3V level translated
16	GPIO12 / CANTXB / MDXB / TZ1	O	CU input. Power stage error signal, driven by protection circuitry. Open-collector signal.
17	-	n.c.	Not connected on the PM50 v3.1
18	-	n.c.	
19	GPIO54 / SPISIMOA / XD25	n.c.	
20	GPIO55 / SPISOMIA / XD24	n.c.	
21	GPIO56 / SPICLKA / XD23	n.c.	
22	GPIO57 / SPISTEA / XD22	n.c.	
23	GND		Ground terminal. All digital signals are referenced to this signal.
24	GND		
25	+5V	O	+5V _{DC} from internal power supply. Connected to J1, J2, J3 connectors. Max. output current : 0.5A.
26	+5V		
27	VREFHI	n.c.	Not connected on the PM50 v3.1

¹ I – input in PM50, output from control unit² CU – control unit (MSK28335 DSC board) connected through the MC-bus with PM50 v3.1³ O – output from PM50, input in control unit

28	VREFLO		Analog ground terminal. Separated from DGND. The connection between these two signals must be done on the CU.
29	ADCIN00	n.c.	Not connected on the PM50 v3.1
30	ADCIN01	n.c.	
31	ADCIN02	n.c.	
32	ADCIN03	n.c.	
33	ADCIN04	O	CU analog input for 3.0V. Value of the motor supply / DC-bus voltage. Gain factor is 54.5 mV/V
34	ADCIN05	O	CU analog input for 3.0V. Value of current in B-phase inverter lower leg, upper shifted with 1.5V. Gain factor is 0.47 V/A
35	ADCIN06	O	CU analog input for 3.0V. Value of current in A-phase inverter lower leg, upper shifted with 1.5V. Gain factor is 0.47 V/A
36	AGND	-	Analog ground terminal. Separated from DGND. The connection between these two signals must be done on the CU.

JP1 - Jumper

The PM50 v3.1 power module provides one jumper JP1 whose position determines the level of the digital input In connected to the J1/J2 MC-Bus:

- JP1 = 1-2, In signal is high (1 logic)
- JP2 = 2-3, In signal is low (0 logic)

MSK28335 DSC board does not use this signal. It was maintained to comply with other control unit devices.

D.7. PM50 v3.1 - Electrical Specifications

Parameter	Conditions	Min.	Typ.	Max.	Units
-----------	------------	------	------	------	-------

DC Input Power

Motor supply		9		36	V
Motor supply current				2.1	Arms
Motor supply current				6.33	Apeak

DC Output supply

Output current on +5V supply to MC Bus J1,J2,J3	For CU		0.5	0.75	A
---	--------	--	-----	------	---

Output Power

Voltage	set by external PWM control	0		36	Vrms
Nominal Motor Power	$V_{in}=36V$, $f_{pwm}=20kHz$, $T_A=40^{\circ}C$			75	W
Nominal Motor Current	$T_A=40^{\circ}C$			1.7	Arms
Overload Motor Current	$V_{in}=36V$, nominal full load current, $f_{pwm}=20kHz$, 0.1 sec. Duration			6.33	Apeak
PWM frequency		0.1	20	100	kHz
Inverter output dead band	Measured at inverter outputs using PWM commands with dead		0.2		μs

	time of 0.5 μ s, $T_A = 25^\circ\text{C}$				
--	---	--	--	--	--

Protections

Output current trip level	$T_C = 25^\circ\text{C}$			7	A_{peak}
Over-voltage trip level		44	47	50	V
Under-voltage trip level	Referring to internal +12V supply, derived from logic supply (+5V)	10		11	V
Short circuit shutdown time	Output terminals shorted (U,V,W)		0.5		ms

Inputs from +5V MC-BUS J1, J2

PWM dead band command	Measured at MC-BUS pins, over operating ambient temperature	0.2	0.5		μ s
High level input voltage	TTL compatible	2			V
Low level input voltage	TTL compatible			0.8	V
High level input current	Compatible with open-collector outputs	0		0	mA
Low level input current				2	mA

+3.3V Inputs from J3

PWM dead band command	Measured at MC-BUS pins, over operating ambient temperature	0.2	0.5		μ s
High level input voltage	3.3V compatible	2		3.6	V
Low level input voltage	3.3V compatible			0.8	V

Analog Outputs

Gain of 3 motor currents feedback	Low-side leg inverter current / 5V bus J1, J2		0.395		V/A
	Low-side leg inverter current / 3.0V bus J3		0.47		V/A
Offset of 3 motor currents feedback on 5V MC Bus	to 5V MC-BUS J1, J2	2.375	2.5	2.625	V
	to 3.0V connector J3	1.49	1.50	1.51	V
Cut-off frequency of 3 motor currents feedback			400		KHz
DC bus voltage feedback gain	5V bus		90.9		mV/V _{BUS}
	3.03V bus		54.5		mV/V _{BUS}
Cut-off frequency of DC bus voltage			730		Hz
Setting time of the 3 motor currents feedback	10% accuracy: 5V bus			1	μ s
	10% accuracy: 3.0V bus			1.1	μ s

Digital Outputs to 5V MC-BUS J1, J2

High level output voltage for Mon/User MC-BUS pin J2/31	Direct link to +5V logic supply		5		V
Low level output voltage for Mon/User MC-BUS pin J2/31	Direct link to ground		0		V
High level output voltage for other signals on MC-BUS J1, J2		2			V
Low level output voltages for other signals on MC-BUS				0.8	V

Digital Outputs to 3.3V BUS J3

High level output voltages		2		3.6	V
Low level output voltages				0.8	V

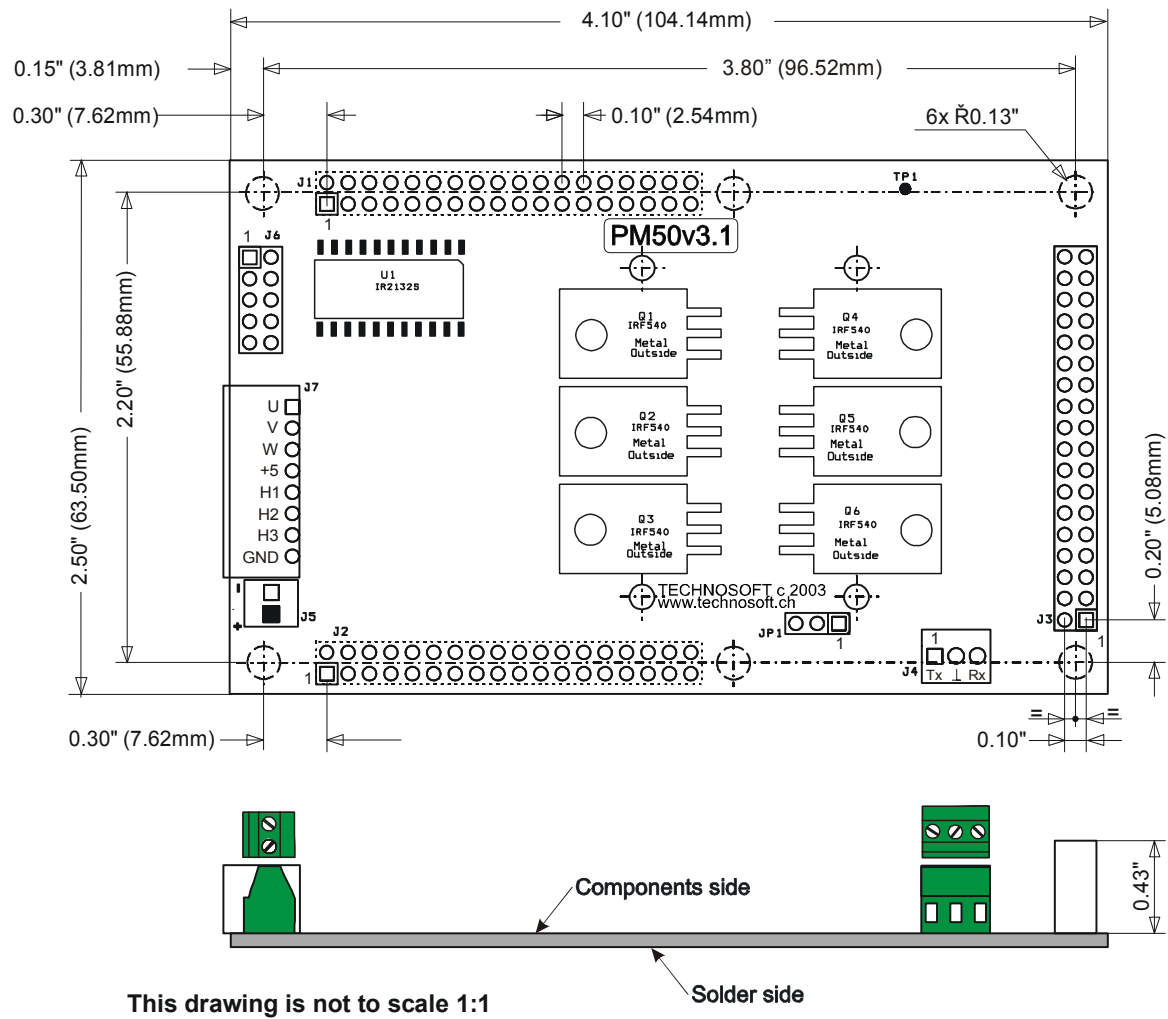
Hall Inputs from motor connector

High level input current	Pull-up resistor to +5V	0		0	mA
Low level input current				1.2	mA

System Environment

Ambient Operating Temp.	90%RH max. (non-condensing)	0		55	°C
-------------------------	-----------------------------	---	--	----	----

D.8. PM50 v3.1 – Mechanical Drawings



This page is empty

Appendix E. MSK Level Adapter v1.0 board

The connection between the MSK28335 DSC board and other Technosoft boards (with the MC bus at 5V) can be done through the MSK Level Adapter board. The MSK Level Adapter board has the following goals:

- To link the MSK28335 DSC board I/O signals from J3 or J4 connectors with the corresponding I/O signals of the MC-Bus connectors J1 and J2
- To perform the 5V-to-3.3V level translation for all the digital signals which are generated by external devices and are connected to the MSK28335 DSC board
- To perform the 5V-to-3.0V level translation for all the analogue signals which are generated by external devices and are connected to the MSK28335 DSC board

The MSK Level Adapter may be present in the MCK28335 Kits if the additional hardware devices (other kit components) request that. If the MSK Level Adapter is not present in the kit it means that all hardware boards can be interconnected without its presence.

E.1. Connecting PM50 V2.x / ACPM750E V3.3 through MSK Level Adapter

J3 Connector (3.3V)		J1 MC-bus Connector		J2 MC-bus Connector		Remarks
Pin	Signal	Pin	Signal	Pin	Signal	
1	+3.3V					
2	+3.3V					
3	PWM1A (o)	3	PWM1(l)			Direct link
4	PWM1B (o)	4	PWM2(l)			Direct link
5	PWM2A (o)	5	PWM3(l)			Direct link
6	PWM2B (o)	6	PWM4(l)			Direct link
7	PWM3A (o)	7	PWM5(l)			Direct link
8	PWM3B (o)	8	PWM6(l)			Direct link
9	GPIO48 / ECAP5 / XD31 (o)	9	ACPM Brake(l)			Direct link
10	GPIO49 / ECAP6 / XD30 (l)			11	Hall 1 (o)	Through CBTD3384
11	GPIO32 / SDAA / EPWMSYNCl / ADCSOCBO (l)			13	Hall 2 (o)	Through CBTD3384
12	GPIO33 / SCLA / EPWMSYNCO / ADCSOCBO (l)			14	Hall 3 (o)	Through CBTD3384
13	GPIO50 / EQEP1A / XD29 (l)	17	Encoder A (o)			Through CBTD3384
14	GPIO51 / EQEP1B / XD28 (l)	18	Encoder B (o)			Through CBTD3384
15	GPIO53 / EQEP1I / XD26 (l)	19	Encoder Z (o)			Through CBTD3384

16	GPIO12 / CANTXB / MDXB / TZ1 (I)	21	Protection (o)			Through CBTD3384
17	GPIO52 / EQEP1S / XD27					
18	GPIO13 / TZ2 / CANRXB / MDRB (o)	10	ACPM Err. Rst (I)			Direct link
19	GPIO54 / SPISIMOA / XD25 (o)			27	SPISIMO (I)	Direct link
20	GPIO55 / SPISOMIA / XD24 (I)			28	SPISOMI (o)	Through CBTD3384
21	GPIO56 / SPICLKA / XD23 (o)			29	SPICLK (I)	Direct link
22	GPIO57 / SPISTE A / XD22 (o)			30	SPISTE (I)	Direct link
23	GND	26 27 28 29	GND	9 10 15 16	GND	Direct link
24	GND	26 27 28 29	GND	9 10 15 16	GND	Direct link
25	+5V	25		21 22		Direct link
26	+5V	25		21 22		Direct link
27	VREFHI					
28	VREFLO	30	VREFLO	7	VREFLO	Direct link
29	ADCINA0					
30	ADCINA1					
31	ADCINA2 (I)	35	ACPM Tacho (o)			5V->3.0V analogue
32	ADCINA3 (I)	36	ACPM Ref (o)			5V->3.0V analogue
33	ADCINA4 (I)	31	ACPM VDC (o)	3	PM50 VDC (o)	5V source selection with jumper, then 5V->3.0V analogue
34	ADCINA5 (I)	34	ACPM I_V (o)	4	PM50 I_V (o)	5V source selection with jumper, then 5V->3.0V analogue
35	ADCINA6 (I)	33	ACPM I_U (o)	5	PM50 I_U (o)	5V source selection with jumper, then 5V->3.0V analogue
36	AGND			6	PM50 I_W (o)	5V->3.0V analogue
				8	VREFHI (o)	Generated on adapter from +5V filtered

Through CBTD3384:

Voltage change for any digital signals of 5V or 3.3V (J1/J2) <-> 3.3V (J3) using IC CBTD3384.

5V->3.0V analogue:

Conversion from 5V -> 3.0V, Uni-directional using Operational Amplifiers.

5V source selection with jumper, then 5V->3.0V analogue:

Selectable conversion (using jumpers) from 5V -> 3.0V, Uni-directional using Operational Amplifiers

There are 3 jumpers on the adapter, which shall be all set to:

- 1-2 to select PM50 signals from J1/J2 to go on J3 (ADCIN)
- 2-3 to select ACPM750 signals from J1/J2 to go on J3 (ADCIN)

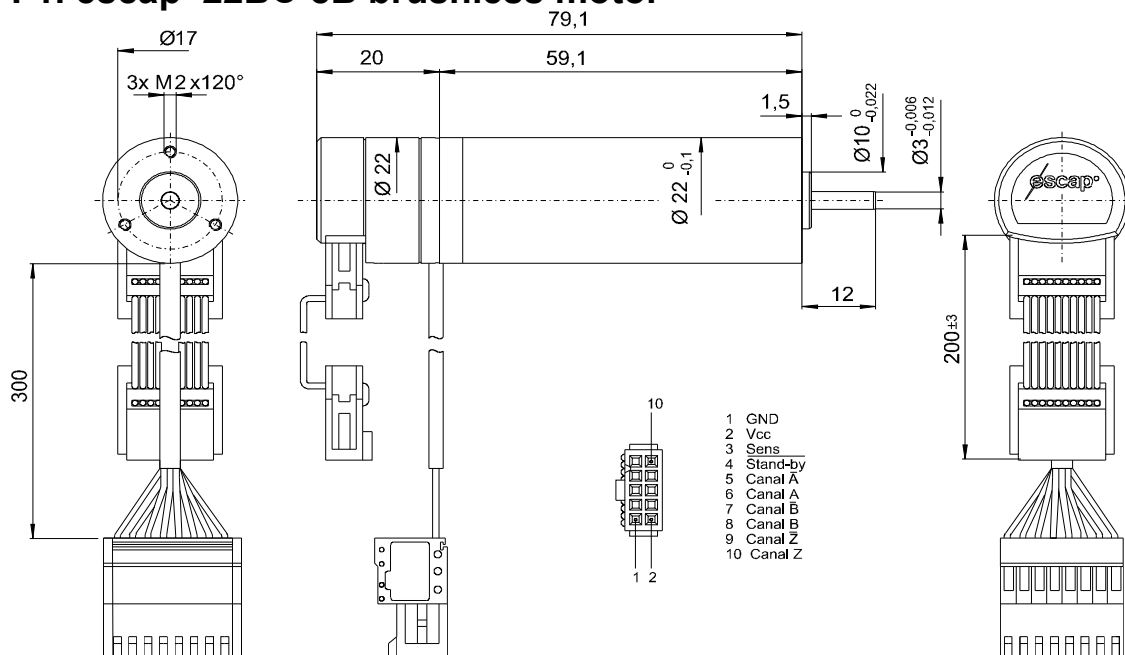
Generated on adapter from +5V filtered:

Voltage reference generator (on the adapter level) for VREFHI (5V) to be used by PM50 or ACPM750.
The MSK28335 DSC board has voltage reference of 3.3V.

This page is empty

Appendix F. Motors Datasheets

F1. escap 22BC-8B brushless motor



Winding type

116

Coil dependent parameters

1	Phase resistance	ohm	7.5
2	Phase inductance	mH	480
3	Back-EMF constant	V/1000 rpm	2.1
4	Torque constant	mNm/A	20

Dynamic parameters

5	Rated voltage	V	10
6	Max. voltage	V	36
7	No-load current	mA	60
8	Max. continuous current	mA	850
9	Max. cont. torque (up to 10'000 rpm)	mNm	17
10	Max. recommended speed 15000	rpm	

Mechanical parameters

11	Peak current	A	3
12	Peak torque	mNm	60
13	Rotor inertia	kgm ² · 10 ⁻⁷	4.6
14	Mechanical time constant	ms	8.6
15	Thermal resistance rotor-body	°C/W	3
16	Thermal res. body-ambient	°C/W	15

Encoder connector

Pin 1	GND
Pin 2	Vcc
Pin 3	Sense
Pin 4	/Stand-By
Pin 5	A-

Pin 6	A+
Pin 7	B-
Pin 8	B+
Pin 9	Z-
Pin 10	Z+

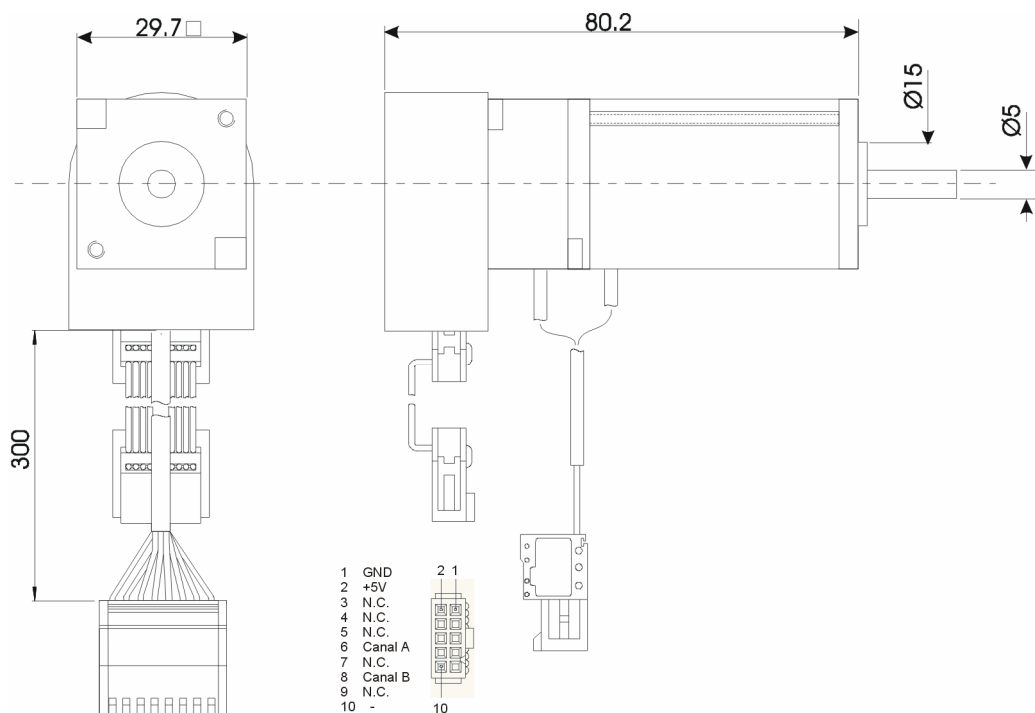
Commutation Sequence

	0°	+60°	+120°	+180°	+240°	+360°
Phase A	+	NC	-	-	NC	+
Phase B	NC	+	+	NC	-	-
Phase C	-	-	NC	+	+	NC
Hall 1	1	1	0	0	0	1
Hall 2	0	1	1	1	0	0
Hall 3	0	0	0	1	1	1

Wire colors

Phase A: Violet	Hall 1: Blue	GND : Yellow
Phase B: Grey	Hall 2: Green	+5V : Red
Phase C: White	Hall 3: Brown	

F2. Pittman 3441 Series brushless motor



Winding type

Wdg #1

Coil dependent parameters

1	Phase resistance	ohm	5.25
2	Phase inductance	mH	0.46
3	Back-EMF constant	V/1000 rpm	2.62
4	Torque constant	mNm/A	25

Dynamic parameters

5	Rated voltage	V	19
6	Max. voltage	V	36
7	No-load current	mA	72
8	Max.continuous stall current	A	3.64
9	Max. cont. torque (up to 10'000 rpm)	mNm	29
10	Max. recommended speed	rpm	8000
11	Peak current	A	1.3
12	Peak torque	mNm	94

Mechanical parameters

13	Rotor inertia	kgm ² . 10 ⁻⁷	9.9
14	Mechanical time constant	ms	8
15	Thermal resi. rotor-ambient	°C/W	8.1

Encoder connector

Pin 1	GND
Pin 2	Vcc
Pin 3	N.C.
Pin 4	N.C.
Pin 5	N.C.

Pin 6	A
Pin 7	N.C.
Pin 8	B
Pin 9	N.C.
Pin 10	N.C.

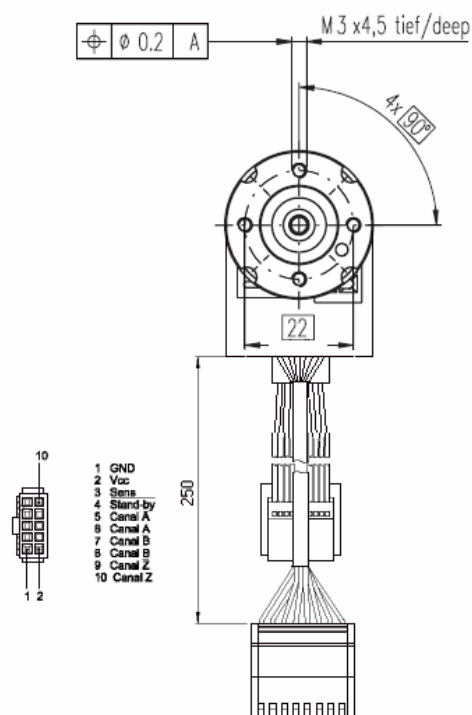
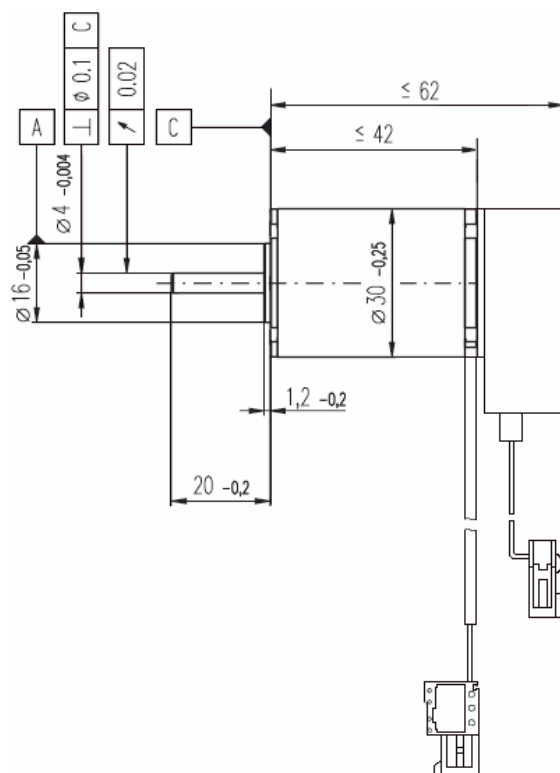
Commutation Sequence:

	0°	+60°	+120°	+180°	+240°	+360°
Phase A	+	+	NC	-	-	NC
Phase B	-	NC	+	+	NC	-
Phase C	NC	-	-	NC	+	+
Hall 1	1	1	0	0	0	1
Hall 2	0	1	1	1	0	0
Hall 3	0	0	0	1	1	1

Wire colors

Phase A: Brown	Hall 1: Grey	GND: black
Phase B: Red	Hall 2: Blue	+5V: violet
Phase C: Orange	Hall 3: white	

F3. Technosoft MBE.300.E500 brushless motor



Coil dependent parameters

1	Phase-phase resistance	ohm	8.61
2	Phase-phase inductance	mH	0.713
3	Back-EMF constant	V/1000 rpm	3.86
4	Torque constant	mNm/A	36.8
5	Pole pairs		1

Dynamic parameters

6	Rated voltage	V	36
7	Max. voltage	V	58
8	No-load current	mA	73.2
9	No-load speed	rpm	9170
10	Max. continuous current (at 5'000 rpm)	mA	913
11	Max. cont. torque (at 5'000 rpm)	mNm	30
12	Max. permissible speed	rpm	15000
13	Peak torque (stall)	mNm	154

Mechanical parameters

14	Rotor inertia	kgm ² · 10 ⁻⁷	11
15	Mechanical time constant	ms	7
16	Thermal resi. housing-ambient		°C/W

17 Thermal resi. winding-housing	°C/W
1.0	

Encoder connector

Pin 1	GND
Pin 2	Vcc
Pin 3	N.C.
Pin 4	N.C.
Pin 5	N.C.

Pin 6	A
Pin 7	N.C.
Pin 8	B
Pin 9	N.C.
Pin 10	N.C.

Commutation Sequence:

	0°	+60°	+120°	+180°	+240°	+360°
Phase A	+	+	NC	-	-	NC
Phase B	-	NC	+	+	NC	-
Phase C	NC	-	-	NC	+	+
Hall 1	1	1	1	0	0	0
Hall 2	0	0	1	1	1	0
Hall 3	1	0	0	0	1	1

Wire colors

Phase A: Red	Hall 1: Yellow	GND: Blue
Phase B: Black	Hall 2: Brown	+5V: Green
Phase C: White	Hall 2: Grey	

This page is empty



T E C H N O S O F T

