SWITCHERGEAR
GEAR TO CONTROL YOUR POWER SWITCHES

SwitcherGear Configuration and Signals

Understanding SwitcherGear Architecture and Signal Chains

APPLICATION NOTE

Table of Contents

Denkinetic Pty Ltd
contact@denki.com.au

Introduction

This document covers the following topics:

1. Overview of the architecture of the SwitcherGear modular controller.
2. How to use the SwitcherGear Configuration Document to trace signal chains.
3. How to trace a signal from the microcontroller to the external power system device, using examples for common signal types, e.g. PWM, analogue signals, etc.

Getting Help

Additional help is available from the SwitcherGear installer (supplied with SwitcherGear hardware) and the website. By default, the installer deploys to `C:/Denkinetic` on your PC.

SwitcherGear Hardware

See folder `SwitcherGear/document` for:

- Reference Manuals for hardware interface modules.
- Reference Manuals for MCU modules.
- Application Notes for real power converter applications.

Supplied with each SwitcherGear controller:

- SwitcherGear Configuration Document.

SwitcherWare Library

See folder `SwitcherGear/document` for :

- Application Note, Using SwitcherWare Library.

See folder `SwitcherWare/<version>/document` for :

- Help file (`.chm` file) for SwitcherWare Library, in document folder of SwitcherWare installation.

Use CCS to import projects from folder `SwitcherWare/<version>/examples` for :

- Usage of SwitcherWare helper classes.
- Real power converter applications, with and without the SwitcherWare framework.

Texas Instruments

See product page for individual microcontroller devices at http://ti.com/

- Datasheet. Overview of device features.
- Technical Reference Manual. Detailed information on operation and usage.
- Other application notes and user guides.

Code Composer Studio http://www.ti.com/tool/CCSTUDIO

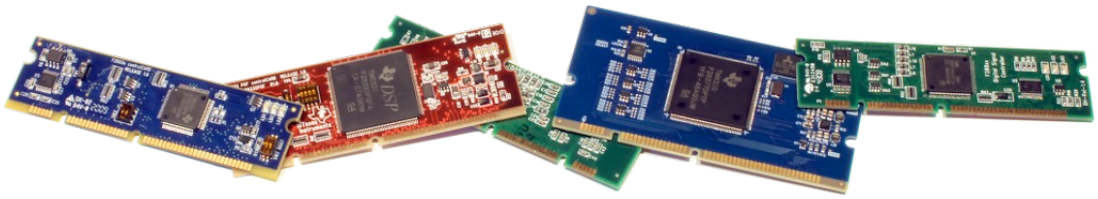
- Download free version of CCS.
- Code generation tools for C2000 microcontrollers.

C2000Ware http://www.ti.com/tool/C2000WARE

- Support resources for C2000 microcontrollers.

SwitcherGear Architecture

Microcontroller Modules



Microcontroller (MCU) modules are plug-in circuit cards in a SwitcherGear controller that execute the user application. The modules provide the features required to implement real-time controllers for power converter systems, including:

- powerful industrial microcontrollers, e.g. TMS320F28377D from Texas Instruments C2000 family
- hardware PWM generation
- high-speed analogue-to-digital converters (ADC)
- power supplies and filtering

The form factor of MCU modules is based on the controlCARD form factor introduced by Texas Instruments. MCU modules have an edge connector interface that mates with the DIM100 connector on the SwitcherGear base. The edge connector contains up to 53 pins (edge finger contacts) for all-purpose digital input/output and 16 pins for analogue inputs (see Figure 1). The remaining pins are for power and the MCU debug interface.

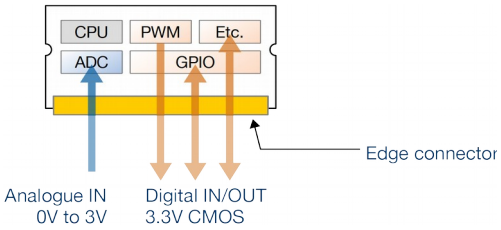


Figure 1: Features of a SwitcherGear MCU module.

The logic level for the digital input/output is 3.3 V CMOS, which is standard for modern MCUs. Digital signals in the MCU can be software assigned to specific pins on the edge connector. This includes PWM signals, LED indicator signals, SPI bus signals, serial communications signals, etc.

The input range of analogue inputs is 0 V to 3 V. The analogue-to-digital converter (ADC) in the MCU can be software configured to automatically capture the analogue inputs. For current and voltage measurements in the power converter, the ADC operation can be synchronised with the PWM generation.

A variety of MCU modules, featuring different MCUs, can be installed in a SwitcherGear controller. This allows developers to select the MCU that best fits the requirements of their particular control application.

Digital Signals

MCUs include many resources for processing digital signals. Simple digital signals are handled as general-purpose input/output (GPIO) digital signal that are under the control of user code. GPIOs are typically used as outputs to control on/off actuators (e.g. LEDs, relays, etc.) and as inputs to read on/off sensors (e.g. switches, etc.). Complex digital signals are handled by specialised hardware peripherals that are integrated onto the silicon of the MCU. There are peripherals for digital signal tasks such as PWM, asynchronous serial communications, incremental encoders, etc.

The digital signals are available on the pins of the MCU package. Each pin includes a multiplexer (MUX) that allows you to select which signal is present on the pin – either a GPIO signal or a selection of peripheral signals. This provides some flexibility in the allocation of pins. The GPIO.MUX number format uniquely identifies the MCU pin and multiplexer setting.

Table 1: Digital signal multiplexing for the GPIO18 signal pin of the TMS320F28377D MCU.

GPIO Pin	MUX ➡	0	1	2	3	4	5
	GPIO.MUX	18.0	18.1	18.2	18.3	18.4	18.5
18	Peripheral	-	SPI A	SCI B	CAN A	-	EPWM 10
	Signal Type	GPIO	CLK	TXD	RX	GPIO	A

There are typically multiple instances of each peripheral type. For example, there can be up to 12 EPWM peripherals (Enhanced PWM) and 4 SCI peripherals (Serial Communications Interface). The different instances are identified by a letter or number suffix (following the convention used by Texas Instruments). For example, SCI B is the second (B) instance of the SCI peripheral type.

A digital signal is uniquely identified by the peripheral that handles it and the signal type. For example, SCI B TXD is the data transmit signal for SCI B.

The same digital signal may be available on multiple pins. For example, on the TMS320F28377D the SCI B TXD signal is available at 9.2, 10.6, 14.2, 18.2, 22.3, 54.6, 70.6, 86.5, 137.6. Not all of these signal pins are available on MCU modules.

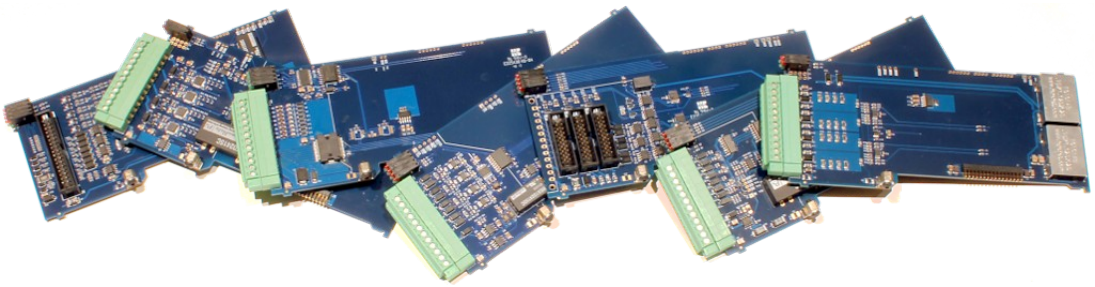
Refer to the Reference Manual for the MCU module for detailed information.

Signals in the Firmware Domain

The physical signals on the pins of the MCU are ultimately transferred into the firmware (software) domain, where they exist as variables/objects that are manipulated by the application code. The transfer between the physical and firmware domains is managed using low-level MCU registers. The SwitcherWare Library provides C++ classes that are used to hide the low-level complexity, allowing the application to access the signals using high-level objects.

Refer to the SwitcherWare Library help and the many example projects.

Hardware Interface Modules



Hardware interface modules are plug-in circuit cards that enable SwitcherGear to interact with power systems. The modules implement functionality such as:

- interfacing to isolated gate-drivers
- interfacing to intelligent power modules (IPM) and converter stacks
- interfacing to power system current & voltage sensors
- interfacing to incremental shaft encoders
- industrial analogue inputs and outputs, e.g. ±10 V, 4 to 20 mA
- industrial digital inputs and outputs, 12 to 24 V
- physical layer for communications protocols, e.g. RS-232, RS-485, Ethernet

Modules share a common hardware form and MCU interface – see Figure 2. The system interface and indicator LEDs vary depending on the functionality of the module. The Reference Manual for each module contains the details for these interfaces, and describes the functionality of the module and how to use it in your application.

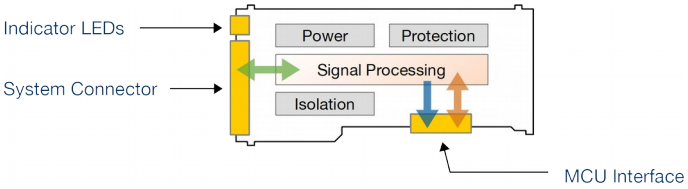


Figure 2: Features of a SwitcherGear hardware interface module.

Modules perform the electrical conversion between the signals of the internal MCU and the signals of the external power system. This can include analogue signal processing, decoding of bus protocols, line drivers for out-going signals, protection against external interference, protection against external faults and isolation to prevent ground loops.

Many modules also include power supplies to power the externally connected devices. For example, the AIN004 module generates +12 V and –12 V power for current sensors and voltage sensors. The CON002 module generates 15 V power for isolated half-bridge gate-drivers.

MCU Interface

The MCU interface is a standardised connector interface for digital and analogue signals that allows the MCU to interact with the hardware interface module. The MCU interface also supplies power to the module – to power the circuitry on the module and for generating power supplies for external devices.

The MCU interface defines 12 pins for all-purpose digital input/output and 4 pins for analogue output (module to MCU). The logic level for the digital pins is 3.3 V CMOS. The range of the analogue pins is defined to be 0 V to 3 V.

The pins of the MCU interface are named as shown in Table 2. The digital pins in the MCU interface have the names D0 to D11. The analogue pins have the names A0 to A3.

Table 2: Naming convention for pins of the MCU interface connector.

MCU Interface Pin		
☐	☐	
	0 … 11	ID number for digital signals
	0 … 3	ID number for analogue signals
D	Digital signal	
A	Analogue signal	

Individual module types make specific use of the digital and analogue pins in order to implement their specific functionality. For example, the AIN004 module uses only the 4 analogue pins to send sensor signals to the MCU. The CON003 module makes use of 10 digital pins (6 PWM input, 1 fault output, 3 SPI bus) and 4 analogue pins (3 phase current sensor, 1 DC link voltage sensor).

Refer to the module Reference Manuals for detailed information about the MCU interface for each type of module.

System Interface

The wiring from the power system connects to the system interface of the modules. This includes wiring from gate-drivers, sensors, communication links, user controls and indicators, etc. The wiring and signals differ for each module type and depend on the features of the module.

Many modules use a 12-way pluggable terminal strip as the system interface connector. Other modules use different connectors to suit specialised requirements.

Only some of the available connections may be used, depending on the external power system. For example, the CON002 module has three header connectors that are used to connect 20-way ribbon cables to industry standard half-bridge gate-drivers. All three headers are used to interface to a 3-phase voltage-source inverter, but only one header would be used for a brake chopper or boost converter.

Refer to the module Reference Manuals for detailed information about the system interface connector for each type of module.

Feature Configuration

Some module types have configurable features. For example, the input range of the AIN004 module can be set between 20 mA and 200 mA, and the mode can be set to unipolar or bipolar.

The settings are typically made using solder jumpers on the circuit board of the module. Initial settings are made at the factory in line with your application's requirements.

Refer to the module Reference Manuals for detailed information about the configurable features of each type of module.

Refer to the SwitcherGear Configuration Document for your SwitcherGear controller for information about how the configuration settings in your controller. Also refer to the rest of this Application Note to understand the SwitcherGear Configuration Document.

Base Unit

SwitcherGear MCU modules and hardware interface modules are installed into a base unit. The B12CC1 base unit has one DIM100 connector to accept a MCU module and 12 module slot connectors to accept hardware interface modules – see Figure 3.

The base unit is responsible for routing the digital and analogue signals between the MCU module and the MCU interfaces of the installed hardware interface modules. The routing is configured at the factory according to your application's requirements.

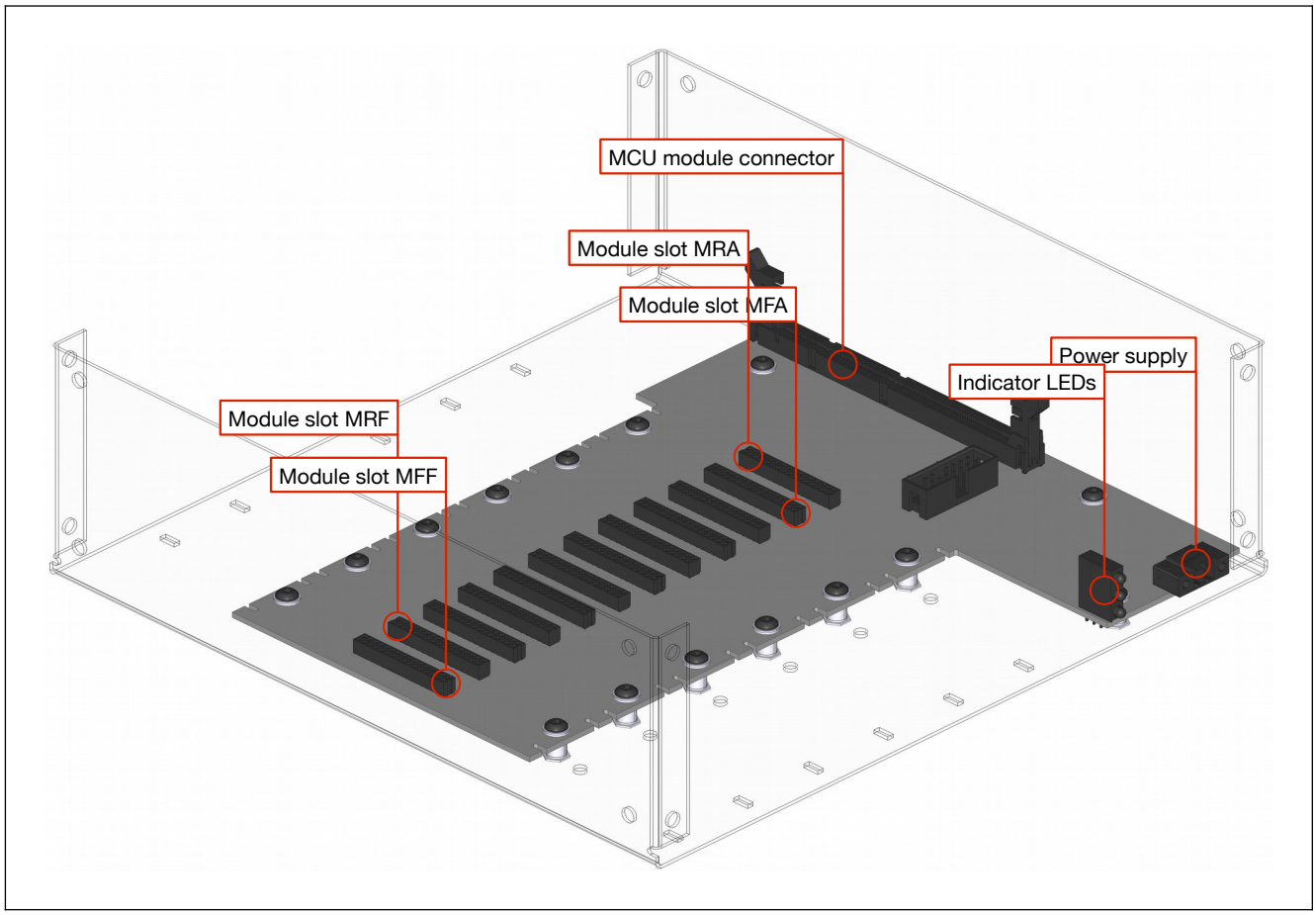


Figure 3: Internal view of the SwitcherGear B12CC1 base unit, showing the MCU module connector and 12 module slot connectors.

Refer to the SwitcherGear Configuration Document for your SwitcherGear controller for detailed information about the signal routing in your controller. Also refer to the rest of this Application Note to understand the routing of signals and the SwitcherGear Configuration Document.

Module Slots

Only some of the module slot connectors may be in use for a particular application. The remaining slots are unused, or can be configured to allow for use with alternative applications or to enable optional features.

The module slots are named as shown in Table 3. For example, slot MRC is the third rear-facing module slot.

Table 3: Naming convention for module slots of SwitcherGear controller base.

Module Slot		
M	☐	☐
		A … F ID letter for front and rear slots
	F Front facing slot **R** Rear facing slot	
M Module slot for SwitcherGear hardware interface module		

Wiring Segregation

The wiring of power converters and associated devices in a power system carries high voltages and currents, and high levels of electrical transients. Other wiring in the system is strictly in the low voltage domain and can be sensitive to interference. Good practice requires that the wiring of the power converter should be segregated from the low voltage wiring.

Module slots can be front facing or rear facing. This feature facilitates the segregation of the power converter wiring from the low voltage wiring.

For a hardware interface module that is installed in a rear facing module slot, the system interface of the module is presented through the rear panel of the base unit. Wiring to these modules is at the rear of the base unit. This is preferred for the wiring of gate-drivers, intelligent power modules, converter stacks and power system current & voltage sensors.

For a hardware interface module that is installed in a front facing module slot, the system interface of the module is presented through the front panel of the base unit. Wiring to these modules is at the front of the base unit. This is preferred for the wiring of incremental encoders, industrial signalling, communications, user controls, etc.

Power Supply

A 24 V control supply is wired to the connector on the front panel of the base unit. This is used to power the MCU module and the hardware interface modules. The same power is used by the hardware interface modules to generate power supplies for external devices connected to them. The single control supply provides power for the SwitcherGear controller and most external devices in the power system, including gate-drivers, sensors, incremental encoders, etc.

Debug Probe

For C2000 development, the XDS110 debug probe plugs into the 14-pin JTAG connector on the front panel of the SwitcherGear controller. The JTAG interface features galvanic isolation.

The SwitcherGear system is designed to be used with any development tool chain for your chosen MCU. Texas Instruments provides Code Composer Studio (CCS) software and extensive support resources for free for code development in C/C++. Alternatively, model-based development and graphical coding tools are available from other vendors.

The SwitcherWare Library can be used with these tools to simplify code access to the MCU and to provides many code resources for power converters.

SwitcherGear Configuration Document

SwitcherGear controllers are modular, with a wide variety of microcontroller and hardware interface options to choose from. Controllers can be configured in many different ways to suit different applications. The configuration for any particular controller is captured in a SwitcherGear Configuration Document, which becomes the reference document for the configuration of that controller.

The document captures the following configuration information:

- The type of the controller base.
- The type of the microcontroller module.
- The type of hardware interface module that can be installed in each slot of the base.
- The internal signal routing between each module and the microcontroller.
- The optional definition of SwitcherWare library objects for MCU signals and SPI buses.
- The external user wiring between each module and the external power system.

You can use the SwitcherGear Configuration Document for your controller to:

- know which microcontroller and hardware interface modules are installed
- determine the signal paths
- allocate software objects to simplify application coding in C/C++
- document how the controller is used in your application

Configuration Code

Each unique configuration of SwitcherGear controller is identified by a unique three letter code and has a corresponding SwitcherGear Configuration Document that describes the configuration.

A particular configuration may be used to implement many SwitcherGear controllers for standard applications, or may be used to implement only one SwitcherGear controller for an unusual application. In every case, the SwitcherGear controller has a label that shows the code of the configuration used to implement the controller and is supplied with the corresponding SwitcherGear Configuration Document.

Editable Document

The SwitcherGear Configuration Document has two forms: an editable ODS document; and a reference PDF document.

The editable ODS document should be opened using the LibreOffice office suite. Download the latest version of LibreOffice for free from

http://www.libreoffice.org/

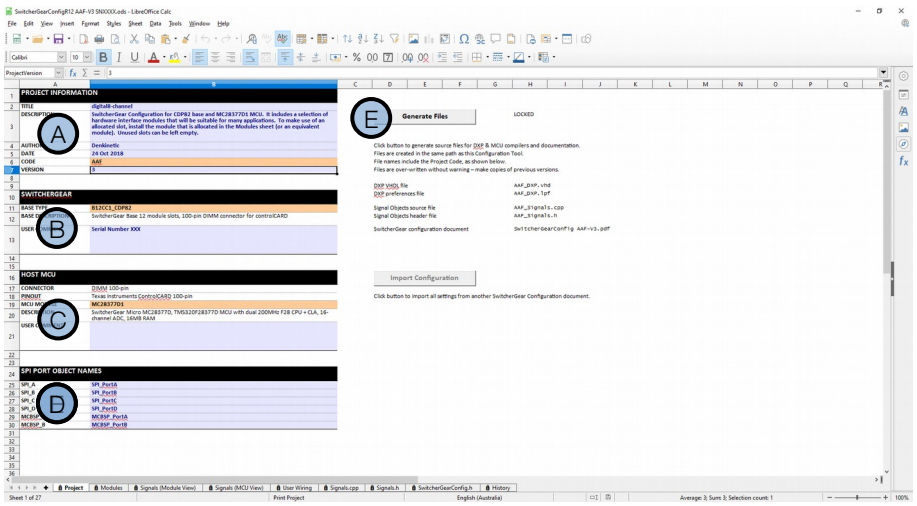
The ODS document has user-editable fields that are highlighted in blue. This allows you to enter comments about the usage and configuration of the controller, and also to define SwitcherWare objects for signals.

The orange fields are set at the factory and cannot be changed by the user.

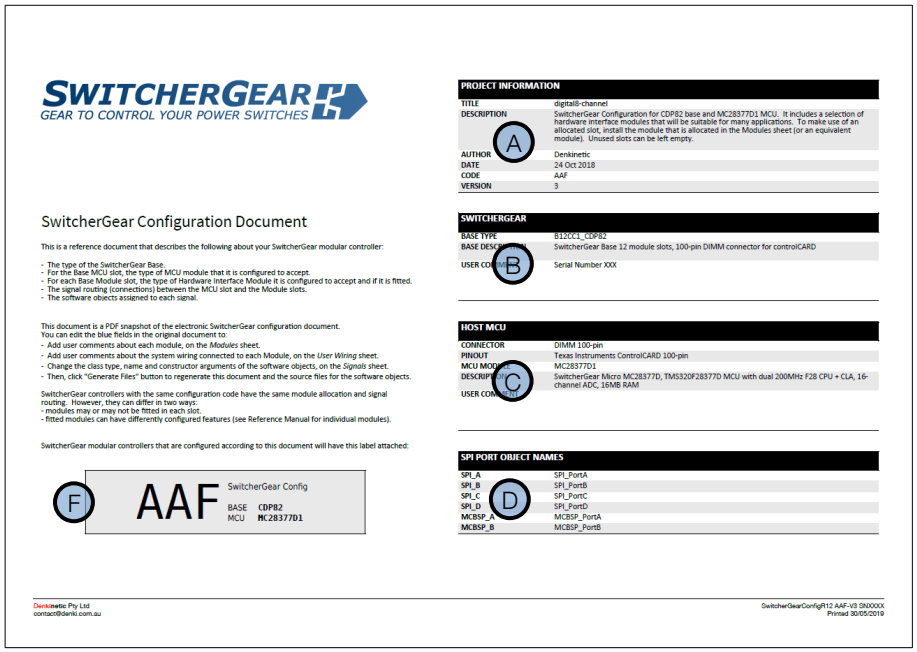
The reference PDF document is created from the ODS document – see section Generating PDF and Source Files. It is intended for printing and archive purposes.

Cover Page

Editable ODS Document



Reference PDF Document



A User description about the application in which the controller is used, and document meta data.

B The type of the SwitcherGear base for this configuration. The description field is automatically filled according to the selected base. The user comment field should include the serial number of the controller. The serial number is marked on the controller.

C The type of the microcontroller (MCU) module for this configuration. The description field is automatically filled according to the selected MCU module.

D The names of automatically constructed SPI bus objects.

E Button to generate the PDF Reference Document and C/C++ source files. Files will be saved using the names shown and any existing files will be over-written without warning. Macros must be enabled to allow this feature.

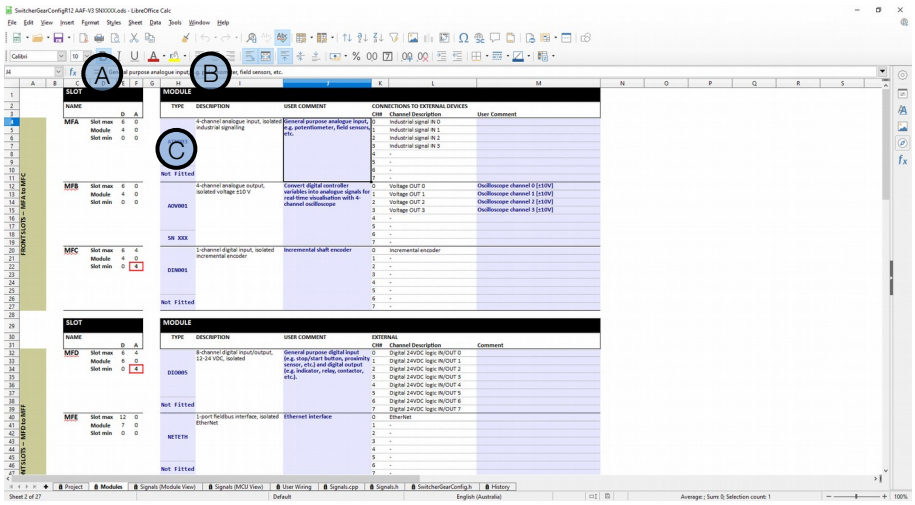
F Representation of the SwitcherGear Configuration Label, showing the three-letter configuration code. The label is attached to every SwitcherGear controller.

Module Usage Pages

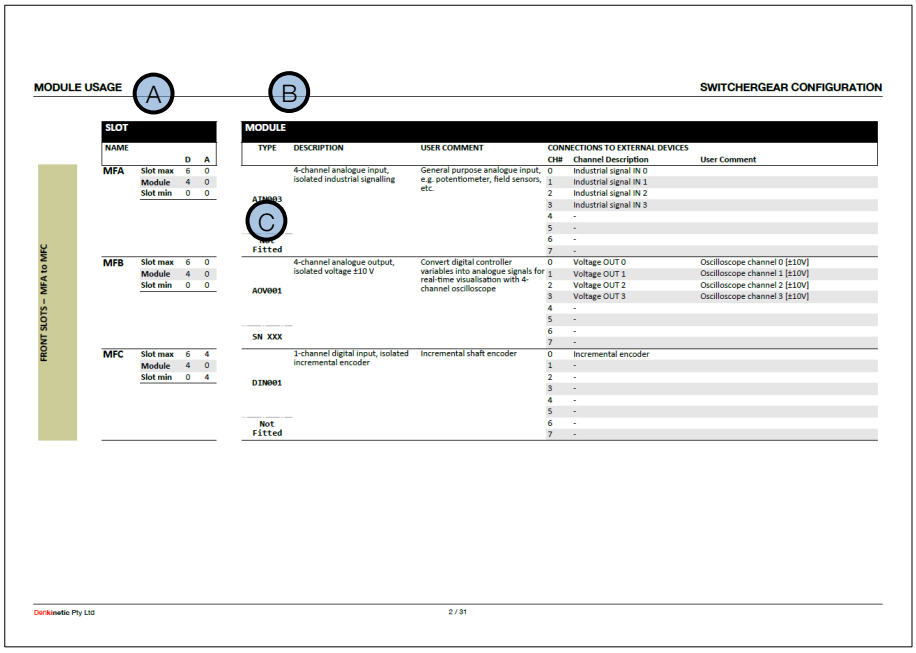
These pages document the type of hardware interface module that can be installed in each module slot. Your SwitcherGear controller may not have all modules installed – slots may be optionally empty. Do not install a different type of module into any slot without first consulting the factory.

Note that you are able to modify the module type. However, because you cannot change the signal routing, you must only change to a module that has compatible signals on its MCU interface. For example, this is possible for some modules that communicate with the MCU using SPI bus.

Editable ODS Document



Reference PDF Document



A The names of the module slots that are available in the selected base. The numbers indicate the number of digital and analogue signals used by the selected module and the min/max number that the module slot can accept.

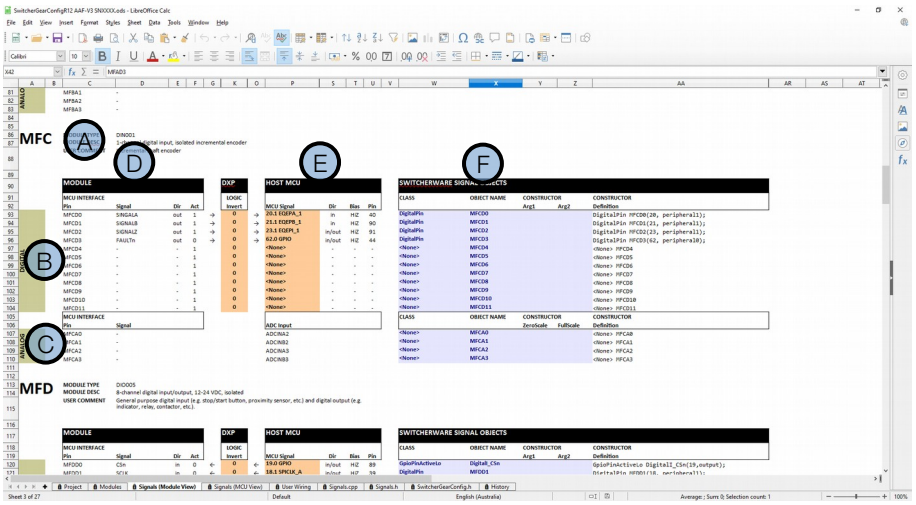
B Details for the modules that are installed in each module slot. The description fields are automatically filled according to the selected module and provide information about the features of the selected module and the signal channels that it implements. The user comment fields allow you to add information about how the modules are used and the settings of the configurable features of the modules.

C The type of the module selected for the slot, optionally including the serial number of the module.

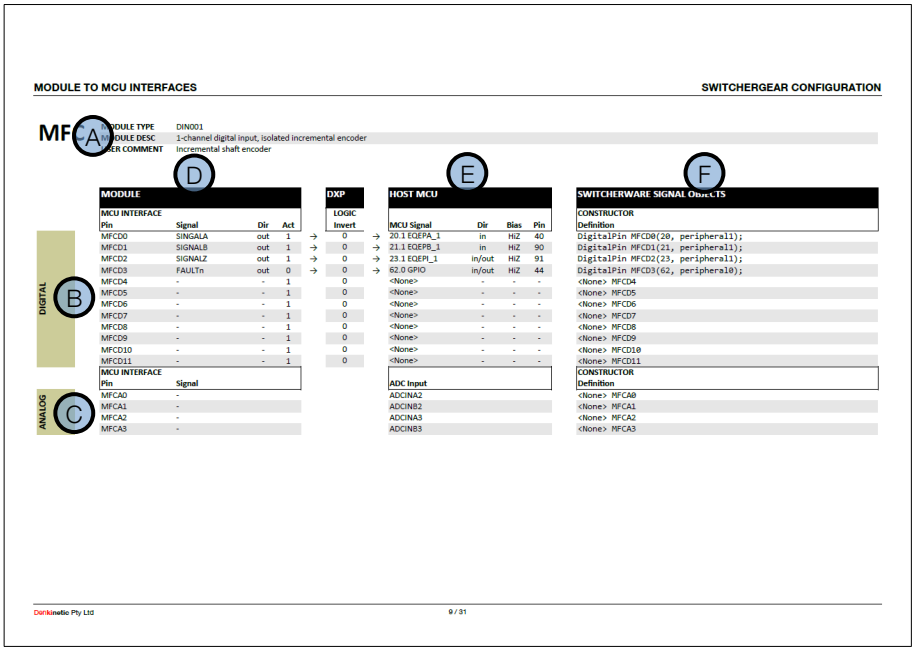
Module to MCU Signal Routing Pages

These pages capture information about the routing of internal signals between the modules and the microcontroller. The digital and analogue signals are grouped by module slot and sorted by MCU interface pin name. Module selection data from the Module Usages pages is used to fill information about the type of module selected for each slot and the signals on this module's MCU interface.

Editable ODS Document



Reference PDF Document



A The name of the module slot and information about the module selected for this slot. This data is taken from the Module Usage pages of the Configuration.

B Digital i/o signals group of the module slot.

C Analogue signals group of the module slot.

D The signals of the MCU interface of the selected module. For each Pin in the module's MCU interface, the Signal column indicates the name of the signal that appears on the pin and the signal direction (as seen from the module). A dash indicates that the module does not make use of the pin. This data is taken from the Reference Manual for the selected module.

E Corresponding signals of the microcontroller. For each Pin in the module's MCU interface, the MCU Signal column indicates the name of the MCU signal that it is routed to, the possible signal directions that the MCU signal can take (as

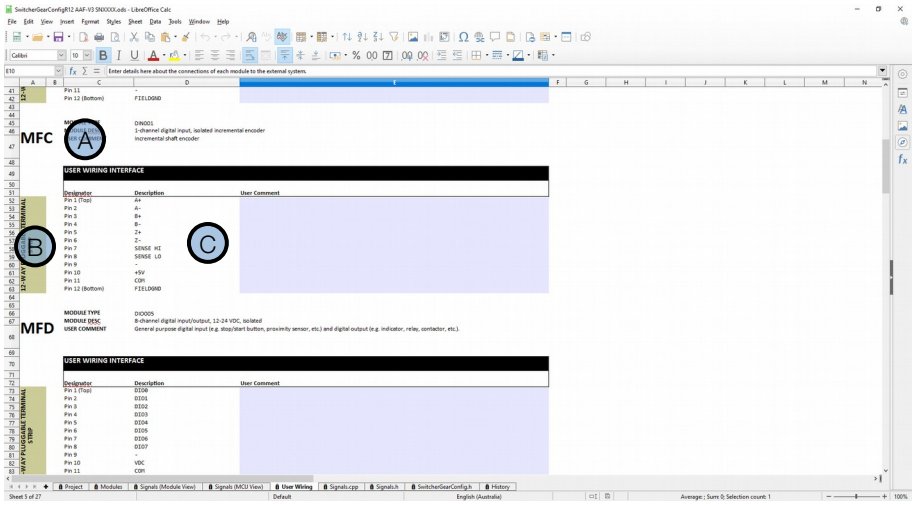
seen from the MCU) and the pin number on the microcontroller module. The orange highlighting indicates that MCU signal routing is factory set and cannot be changed by the user.

F Corresponding SwitcherWare objects. For each Pin in the module's MCU interface, you can modify the type of SwitcherWare object that is allocated to the signal, the name of the object and any required constructor arguments for the object. The Constructor Definition column provides a preview of the object constructor.

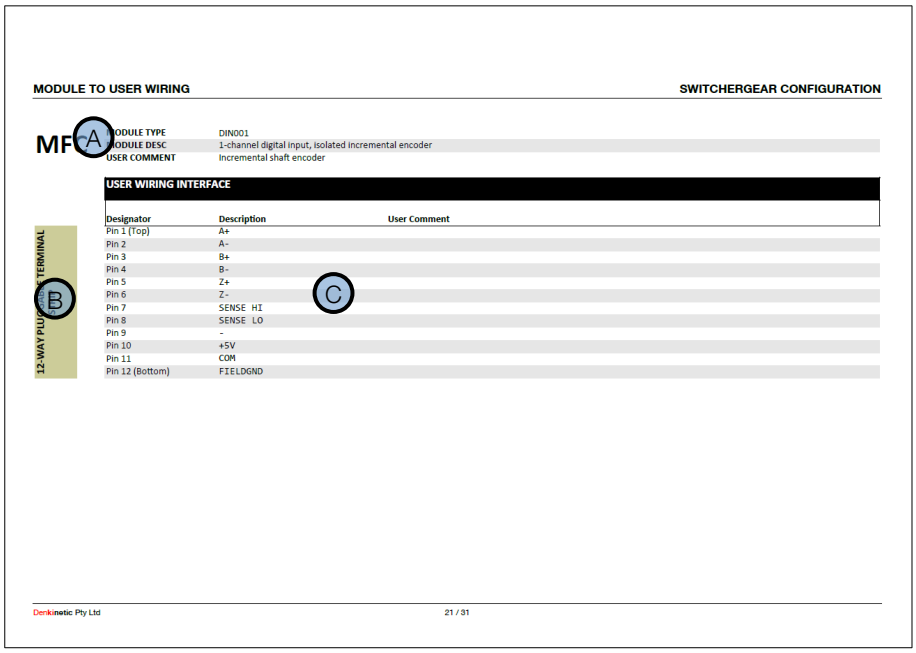
Module User Wiring Pages

These pages capture information about the wiring to the system connector of each module.

Editable ODS Document



Reference PDF Document



A The name of the module slot and information about the module selected for this slot. This data is taken from the Module Usage pages of the Configuration.

B The type of connector used for the system connection on the selected module.

C The wiring connections are listed in order of system connector designator of the selected module. Designators vary depending on the type of system connector used by the selected module. Many modules use 12-way pluggable terminal headers and the designators are the numbered pins of the connector. Where ribbon cables are used, one designator is assigned to each pin header connector. For each wiring designator, the Description column indicates the name of the signal that appears there. This data is taken from the Reference Manual for the selected module.

Generating PDF and Source Files

The reference PDF document is generated by a macro in the editable ODS document. The macro also generates C/C++ source/ header files that can be used with the SwitcherWare library in your application project.

Click on the Generate Files button on the cover page of the ODS document. The files will be created in the same folder as the ODS document with file names as shown in the preview below the button. Previously generated files with the same name will be over-written without any warning.

Macros must be be enabled for the ODS document. Select the Tools menu item, navigate to LibreOffice > Security, press the Macro Security button and select Medium security level. You will be asked to allow macros when the ODS document is opened.

Using Source Files in Your Project

You can configure your project to use the automatically generated source files:

- Identify the ODS document with three-letter Configuration code that matches your SwitcherGear controller.
- Inside your CCS project, make a folder called SwitcherGear Configuration.
- Place a copy of the ODS document into the SwitcherGear Configuration folder.
- Edit the copied ODS document to ensure that objects are allocated for the signals that you want to access in your application code. Check that the class type, name and constructor arguments are correct. The class type for other signals should be set to <None>.
- Generate the source/ header files.
- Add the SwitcherGear Configuration folder to the include search paths for your CCS project. Right click on the project in the Project Explorer pane and select the Properties menu item, navigate to Build > C2000 Compiler > Include Options. Click on the + symbol to add a new #include search path. Click on Workspace and select the folder inside your project.
- In your own source files that reference the signal objects, add the following line at the top

```
#include “SwitcherGearConfig.h”
```

- Add the SwitcherWare Library to your project – see separate application note Install SwitcherWare Library.
- Build your project.

In the above scheme, the generated source file is part of the project because it is located inside the project folder. The build processor will detect the `.cpp` source file, compile it and linked it into the executable file.

The example projects supplied with the SwitcherWare Library use a modified scheme. A single ODS document is kept in a common folder that is separate to the project folders. The generated source file (in the same folder and ending in `.cpp`) is added to a project as a linked file. The common folder is added to the include search paths of the project. Multiple projects can link to the single source file. Changes to the ODS document and the generated source file will propagate to all projects that link to it. This is useful when multiple projects will be developed concurrently on the same controller hardware.

Signal Chains

In the examples of the following sections, signal chains are traced for some common types of signals. Table 4 lists the components that make up the signal chain – see section SwitcherGear Architecture for a description. The table indicates which documentation to refer to in order to find out about each component in the signal chain.

The following signal chain examples are for a SwitcherGear controller with configuration code AAF.

Each signal chain example provides:

- a description of the signal chain
- an illustration showing the signal chain from the external device via the module to the MCU
- the signal routing of the signal chain from the AAF SwitcherGear Configuration Document
- a table (like Table 4) that lists each point in the signal chain, e.g. MCU signal, module pin, firmware object, etc.

Table 4: Documentation associated with various parts of the signal chain.

			SwitcherGear Reference Manual	SwitcherGear Configuration Document	SwitcherWare Library Help
SIGNAL CHAIN	Firmware	Application helper object			✓
		Pin level object		✓	✓
	MCU Module	MCU function description	(Manufacturer Documentation)		
		MCU signal	✓	✓	
	Base Unit	MCU module connector pin		✓	
		Signal routing direction		✓	
		Module slot connector pin		✓	
	Hardware Interface Module	MCU interface signal	✓	✓	
		Functional description	✓		
		System connector	✓	✓	
	External Device	Functional description	(Manufacturer Documentation)		

PWM Digital Signal

Gate-drivers are connected to the SwitcherGear controller using a CONxxx series module. These modules provide power to the gate-drivers, drive the gate signals with a robust line driver and accept a fault signal. The modules have no isolation because isolated gate-drivers are normally used. In the AAF configuration, a CON002 module is installed in module slot MRE. Here we consider the signal chain for the gate PWM signals for a half-bridge gate-driver.

The signal chain between the external gate-drivers and the CON002 module is shown in Figure 4 by the green arrow. A standard half-bridge gate driver, such as Power Integrations 2SP0115 or Semikron Skyper12C, is connected by a 20-way ribbon cable to the HB1 system connector of the module.

The signal chain between the interface module and the MCU is shown in Figure 4 by the orange arrow. This part of the signal chain is detailed in the Module Signal Routing for slot MRE, as shown in Figure 5. The bottom and top gate signals in the MCU interface for the HB1 channel are HB1CHA and HB1CHB. These signals are routed from the EPWM1 peripheral signals in the MCU. The allocation of a `DigitalPin` objects for these signals configures the GPIO multiplexer to give control of these MCU pins to the EPWM1 peripheral.

The MCU has multiple EPWM peripherals that can generate high-resolution, complementary PWM signals with dead-time. The peripheral is controlled by a large number of low-level registers. SwitcherGear helper classes hide this low-level complexity and allow you to operate half-bridges and 3-phase converters using the high-level object.

You can see in Figure 5 that this module has two other pairs of PWM signals generated by the EPWM2 and EPWM3 peripherals. This module can drive a total of 3 external gate-drivers that control 3 half-bridge power modules. You can control them as buck or boost converters (either independently or interleaved), using the `HB1Sym_F2837xD` helper class. Or you can control them together as a 3-phase voltage-source inverter, using the `HB3Sym_F2837xD` helper class to easily generate 3-phase SVM or sine PWM.

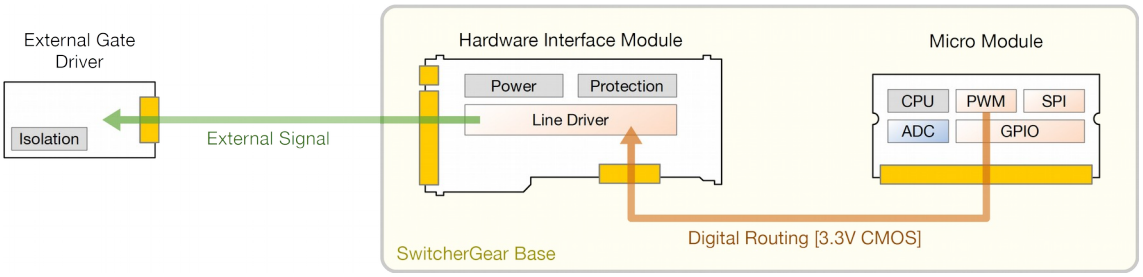


Figure 4: Typical signal chain for a PWM gate-driver digital signal.

MRE

MODULE TYPE	CON002
MODULE DESC	3-phase converter interface, Concept half-bridge gate driver
USER COMMENT	Connect to 3 half-bridge gate drivers of 3-phase VSI. Use adaptor kit for connection to Semikron SemiTeach IGBT. Half-bridges may be part of any other converter topology, e.g. DC-DC buck, boost, etc.

DIGITAL

MODULE					DXP		HOST MCU				SWITCHERWARE SIGNAL OBJECTS
MCU INTERFACE					LOGIC						CONSTRUCTOR
Pin	Signal	Dir	Act		Invert		MCU Signal	Dir	Bias	Pin	Definition
MRED0	HB1CHA	in	1	←	0	←	00.1 EPWM_1A	out	HiZ	23	`DigitalPin MRED0(0, peripheral1);`
MRED1	HB1CHB	in	1	←	0	←	01.1 EPWM_1B	out	HiZ	73	`DigitalPin MRED1(1, peripheral1);`
MRED2	HB1TRIPn	out	0	→	0		<None>	-	-	-	<None> MRED2
MRED3	HB2CHA	in	1	←	0	←	02.1 EPWM_2A	out	HiZ	24	`DigitalPin MRED3(2, peripheral1);`
MRED4	HB2CHB	in	1	←	0	←	03.1 EPWM_2B	out	HiZ	74	`DigitalPin MRED4(3, peripheral1);`
MRED5	HB2TRIPn	out	0	→	0		<None>	-	-	-	<None> MRED5
MRED6	HB3CHA	in	1	←	0	←	04.1 EPWM_3A	out	HiZ	25	`DigitalPin MRED6(4, peripheral1);`
MRED7	HB3CHB	in	1	←	0	←	05.1 EPWM_3B	out	HiZ	75	`DigitalPin MRED7(5, peripheral1);`
MRED8	HB3TRIPn	out	0	→	0		<None>	-	-	-	<None> MRED8
MRED9	HB123TRIPn	out	0	→	0	→	61.0 GPIO	in/out	HiZ	80	`DigitalPin MRED9(61, peripheral0);`
MRED10	-	-	1		0		<None>	-	-	-	<None> MRED10
MRED11	-	-	1		0		<None>	-	-	-	<None> MRED11

ANALOG

MCU INTERFACE	
Pin	Signal
MREA0	-
MREA1	-
MREA2	-
MREA3	-

Figure 5: Excerpt of SwitcherGear Configuration Document (PDF page 17) showing signal chain for module in slot MRE.

Table 5: Signal chain for module pins MRED0-1 – digital PWM signals for an external gate-driver.

SIGNAL CHAIN	Firmware	Application helper object	`HB3Sym_F2837xD -`
		Pin level object	`DigitalPin MRED0, MRED1`
	MCU Module	MCU functional description	Peripheral EPWM 1
		MCU signal	00.1 EPWM_1A, 01.1 EPWM_1B
	Base Unit	MCU module connector pin	23, 73
		Signal routing direction	↓
		Module slot connector pin	MRED0, MRED1
	Hardware Interface Module	MCU interface signal	HB1CHA, HB1CHB
		Functional description	15 V line driver, power supply, fault
		System connector	Ribbon cable HB1
	External Device	Functional description	Half-bridge gate-driver

Sensor Analogue Signal

Current and voltage sensors are typically connected to the SwitcherGear controller using a AIN004 module. This module provides power for the sensor, and converts the sensor output to a voltage and scales it according to jumper settings. The module has no isolation so the sensors must provide safe separation from the live circuits. In the AAF configuration, a AIN004 module is installed in module slot MRB. Here we consider the signal chain for a current sensor connected to channel 0 of the module.

The signal chain between the external gate-drivers and the AIN004 module is shown in Figure 6 by the green arrow. A standard current-output sensor, such as a LEM or SwitcherGear current sensor, is connected by 3 wires to the channel 0 pins of the system connector of the module.

The signal chain between the interface module and the MCU is shown in Figure 6 by the blue arrow. This part of the signal chain is detailed in the Module Signal Routing for slot MRB, as shown in Figure 7. The analogue sensor signal in the MCU interface for channel 0 is VOUT0. This signal is routed to the ADCINC3 pin of the MCU. The signal has been allocated a `AinPinScaled` object named `Iu`.

The user comments for channel 0 of the AIN004 module indicate that the channel is to be configured for bipolar input mode and a range of 20 mA. Therefore, the input range is -20 mA to +20 mA. The constructor arguments for the signal object define the corresponding output range to be -20 A to 20 A. This requires the use of a current sensor with a gain of 1 mA/A, such as the SwitcherGear SNI005 current sensor.

The signal object must be registered with the helper object, which will configure the low-level registers to perform automatic conversions of the analogue signal. After calling `Iu.Update()`, the scaled result is available in the member variable `Iu.outScaled`.

We recommend that you validate the correctness of the signal chain, especially because the user comments may not align with the actual channel settings. Simply apply know currents to the sensor and confirm that the values in signal object agree.

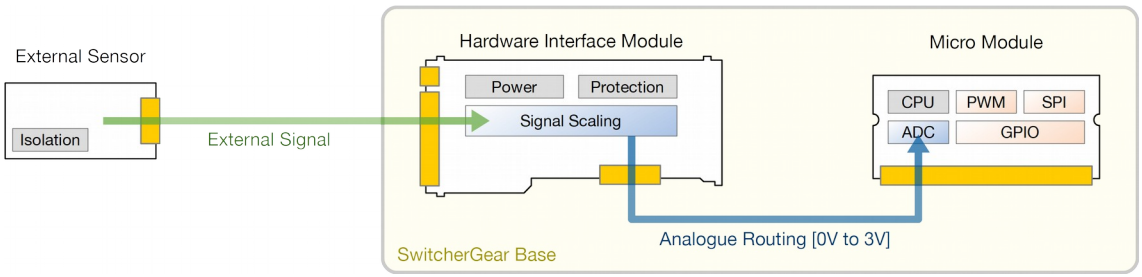


Figure 6: Typical signal chain for a sensor analogue signal.

MRB

MODULE TYPE	AIN004
MODULE DESC	4-channel analogue input, sensor current 20 to 200mA
USER COMMENT	Connect up to 4 sensors for measurement of power system voltages and currents. Suitable sensors include SNV005, SNI005, etc.

DIGITAL

MODULE MCU INTERFACE Pin	Signal	Dir	Act	DXP LOGIC Invert	HOST MCU MCU Signal	Dir	Bias	Pin	SWITCHERWARE SIGNAL OBJECTS CONSTRUCTOR Definition
MRBD0	-	-	0	0	<None>	-	-	-	<None> MRBD0
MRBD1	-	-	0	0	<None>	-	-	-	<None> MRBD1
MRBD2	-	-	0	0	<None>	-	-	-	<None> MRBD2
MRBD3	-	-	0	0	<None>	-	-	-	<None> MRBD3
MRBD4	-	-	0	0	<None>	-	-	-	<None> MRBD4
MRBD5	-	-	0	0	<None>	-	-	-	<None> MRBD5
MRBD6	-	-	0	0	<None>	-	-	-	<None> MRBD6
MRBD7	-	-	0	0	<None>	-	-	-	<None> MRBD7
MRBD8	-	-	0	0	<None>	-	-	-	<None> MRBD8
MRBD9	-	-	0	0	<None>	-	-	-	<None> MRBD9
MRBD10	-	-	0	0	<None>	-	-	-	<None> MRBD10
MRBD11	-	-	0	0	<None>	-	-	-	<None> MRBD11

ANALOG

MCU INTERFACE Pin	Signal	ADC Input	CONSTRUCTOR Definition
MRBA0	VOUT0	ADCINC3	AinPinScaled Iu(ADCINC3, -20, 20);
MRBA1	VOUT1	ADCIND1	AinPinScaled Iv(ADCIND1, -20, 20);
MRBA2	VOUT2	ADCINC2	AinPinScaled Iw(ADCINC2, -20, 20);
MRBA3	VOUT3	ADCIND0	AinPinScaled Vdc_link(ADCIND0, 0.000, 1000.000);

Figure 7: Excerpt of SwitcherGear Configuration Document (PDF page 14) showing signal chain for module in slot MRB.

Table 6: Signal chain for module pin MRBA0 – an analogue signal from a current sensor.

SIGNAL CHAIN	Firmware	Application helper object	`ADC_F2837xD theF2837xDADC`
		Pin level object	`AinPinScaled Iu`
	MCU Module	MCU functional description	Peripheral ADC C
		MCU signal	ADCINC3
	Base Unit	MCU module connector pin	67
		Signal routing direction	↑
		Module slot connector pin	MRBA0
	Hardware Interface Module	MCU interface signal	VOUT0
		Functional description	Sensor input channel 0, bipolar 20 mA
		System connector	Pin 2 (sensor power on pins 1, 3)
	External Device	Functional description	SNI005 current sensor

SPI Bus

SPI is a synchronous serial protocol for transferring data between integrated circuits. The bus consist of four signals – a clock, two data lines and a chip select signal. In the AAF configuration, a AOV004 module is installed in module slot MFB. Here we consider the signal chain for the SPI bus that allows the MCU to interact at a high-level with the module.

The signal chain between the interface module and the MCU is shown in Figure 8 by the orange arrow. This part of the signal chain is detailed in the Module Signal Routing for slot MFB, as shown in Figure 9, and comprises only the signals of the SPI bus.

The four MCU signals here are for a multi-channel buffered serial port (MCBSP), which will be operated in SPI emulation mode. The allocation of a `DigitalPin` objects for these signals configures the GPIO multiplexer to give control of these MCU pins to the MCBSP B peripheral. In the case of a AOV001 module, serial data is transferred only from the MCU to the module, so the SOMI (slave out, master in) signal object is not defined.

The SwitcherWare helper class for the AOV001 module is ModuleAOV001. When it is initialised, it requires a reference to the MCBSP peripheral. The state machine of the MCBSP peripheral operates the SPI bus, including the chip select pin, to transfer serial data to the module.

The signal chain between the external device and the AOV001 module is shown in Figure 6 by the green arrow. These signals are un-related to the digital signals of the SPI bus – they have been transformed by the SPI logic in the module. For

the AOV001 module, the SPI logic performs a digital-to-analogue conversion. Wiring from the module's system connector carries the analogue output signals to the external device, which could be an oscilloscope or other process controller.

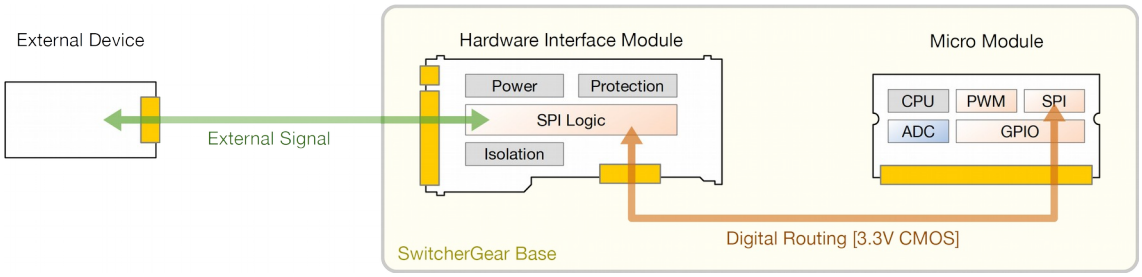


Figure 8: Typical signal chain for a digital SPI bus.

MFB

MODULE TYPE	AOV001
MODULE DESC	4-channel analogue output, isolated voltage ±10 V
USER COMMENT	Convert digital controller variables into analogue signals for real-time visualisation with 4-channel oscilloscope

DIGITAL

MODULE					DXP		HOST MCU				SWITCHERWARE SIGNAL OBJECTS
MCU INTERFACE					LOGIC						CONSTRUCTOR
Pin	Signal	Dir	Act		Invert		MCU Signal	Dir	Bias	Pin	Definition
MFBD0	CSn	in	0	←	0	←	27.3 MFSX_B	in/out	HiZ	86	DigitalPin AOV001_CSn(27, peripheral3);
MFBD1	SCLK	in	1	←	0	←	26.3 MCLKX_B	in/out	HiZ	36	DigitalPin MFBD1(26, peripheral3);
MFBD2	SIMO	in	1	←	0	←	24.3 MDX_B	out	HiZ	35	DigitalPin MFBD2(24, peripheral3);
MFBD3	SOMI	out	1	→	0	→	25.3 MDR_B	in	HiZ	85	<None> MFBD3
MFBD4	-	-	1		0		<None>	-	-	-	<None> MFBD4
MFBD5	-	-	1		0		<None>	-	-	-	<None> MFBD5
MFBD6	-	-	1		0		<None>	-	-	-	<None> MFBD6
MFBD7	-	-	1		0		<None>	-	-	-	<None> MFBD7
MFBD8	-	-	1		0		<None>	-	-	-	<None> MFBD8
MFBD9	-	-	1		0		<None>	-	-	-	<None> MFBD9
MFBD10	-	-	1		0		<None>	-	-	-	<None> MFBD10
MFBD11	-	-	1		0		<None>	-	-	-	<None> MFBD11

ANALOG

MCU INTERFACE	
Pin	Signal
MFBA0	-
MFBA1	-
MFBA2	-
MFBA3	-

Figure 9: Excerpt of SwitcherGear Configuration Document (PDF page 8) showing signal chain for module in slot MFB.

Table 7: Signal chain for module pins MFBD1-3 – a digital SPI bus for analogue output.

SIGNAL CHAIN	Firmware	Application helper object	`ModuleAOV001 -`
		Pin level object	`DigitalPin AOV001_CSn, MFBD1, MFBD2 SPIbus MCBSP_PortB`
	MCU Module	MCU functional description	Peripheral MCBSP B
		MCU signal	27.3 MFSX_B, 26.3 MCLKX_B, 24.3 MDX_B
	Base Unit	MCU module connector pin	86, 36, 35
		Signal routing direction	↕
		Module slot connector pin	MFBD0, MFBD1, MFBD2
	Hardware Interface Module	MCU interface signal	CSn, SCLK, SIMO
		Functional description	Analogue output, ±10 V
		System connector	Pins 1-8
	External Device	Functional description	4-channel oscilloscope

General-Purpose Digital Signal

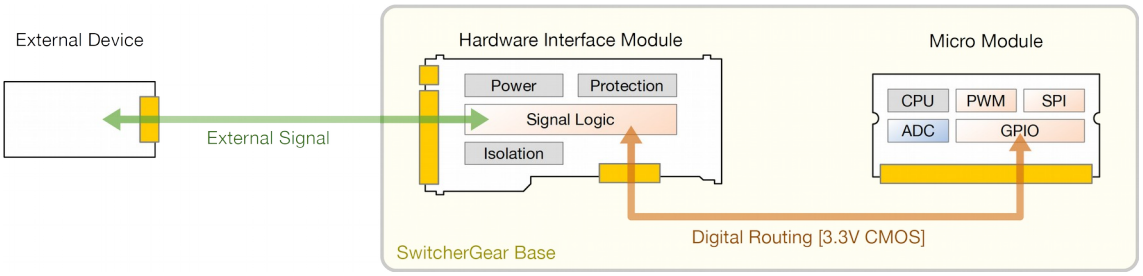


Figure 10: Typical signal chain for a general-purpose digital signal.

BASE

BASE TYPE	B12CC1_CDP82
BASE DESCRIPTION	SwitcherGear Base 12 module slots, 100-pin DIMM connector for controlCARD
USER COMMENT	

	BASE PERIPHERALS				DXP		HOST MCU				SWITCHERWARE SIGNAL OBJECTS
	MCU INTERFACE Pin	Signal	Dir	Act	LOGIC Invert		MCU Signal	Dir	Bias	Pin	CONSTRUCTOR Definition
TEST POINT	BTPD0	TP0	in	1	0	←	34.1 OUTPUTXBAR1	out	HiZ	46	DigitalPin BTPD0(34, peripheral1);
	BTPD1	TP1	in	1	0		<None>	-	-	-	<None> BTPD1
	BTPD2	TP2	in	1	0		<None>	-	-	-	<None> BTPD2
	BTPD3	TP3	in	1	0		<None>	-	-	-	<None> BTPD3
	MCU INTERFACE Pin	Signal	Dir	Act	LOGIC Invert		MCU Signal	Dir	Bias	Pin	CONSTRUCTOR Definition
TEMP	BTSD0	CSn	in	0	0	←	42.0 GPIO	in/out	HiZ	45	GpioPinActiveLo Temp_CSn(42,output);
	BTSD1	SCLK	in	1	0	←	18.1 SPICLK_A	in/out	HiZ	39	DigitalPin BTSD1(18, peripheral1);
	BTSD2	SOMI	out	1	0	→	17.1 SPISOMI_A	in/out	HiZ	88	DigitalPin BTSD2(17, peripheral1);
	MCU INTERFACE Pin	Signal	Dir	Act	LOGIC Invert		MCU Signal	Dir	Bias	Pin	CONSTRUCTOR Definition
NON-VOLATILE MEMORY	BFRD0	CSn	in	0	0	←	103.6 SPISTE_C	in/out	HiZ	54	GpioPinActiveLo FRAM_CSn(103,output);
	BFRD1	SCLK	in	1	0	←	102.6 SPICLK_C	in/out	HiZ	4	DigitalPin BFRD1(102, peripheral6);
	BFRD2	SIMO	in	1	0	←	100.6 SPISIMO_C	in/out	HiZ	3	DigitalPin BFRD2(100, peripheral6);
	BFRD3	SOMI	out	1	0	→	101.6 SPISOMI_C	in/out	HiZ	53	DigitalPin BFRD3(101, peripheral6);
	BFRD4	HOLDn	in	0	0	←	43.0 GPIO	in/out	HiZ	95	GpioPinActiveLo FRAM_HOLDn(43,output);
	MCU INTERFACE Pin	Signal	Dir	Act	LOGIC Invert		MCU Signal	Dir	Bias	Pin	CONSTRUCTOR Definition
LED	BULD0	LED RED	in	0	0	←	58.0 GPIO	in/out	HiZ	18	GpioPinActiveLo RedLED(58,output);
	BULD1	LED GRN	in	0	0	←	59.0 GPIO	in/out	HiZ	68	GpioPinActiveLo GrnLED(59,output);

Figure 11: Excerpt of SwitcherGear Configuration Document (PDF page 6) showing signal chain for the base unit.

Table 8: Signal chain for front panel red LED – a general-purpose digital signal.

SIGNAL CHAIN	Firmware	Application helper object	
		Pin level object	`GpioPinActiveLo RedLED`
	MCU Module	MCU functional description	GPIO
		MCU signal	58.0 GPIO
	Base Unit	MCU module connector pin	18
		Signal routing direction	↓
		Module slot connector pin	BULD0
	Hardware Interface Module	MCU interface signal	-
		Functional description	-
		System connector	-
	External Device	Functional description	Front panel red LED

There are other examples of general-purpose digital signals in Figure 11, notably the SPI bus chip select signals for the temperature sensor and non-volatile memory. Unlike the previous SPI bus example, the chip select pins for these SPI devices are not controlled directly by the MCU peripheral hardware. This is to allow multiple SPI devices to be connected

to the same SPI bus, each with its own chip select pin. The SwitcherGear helper classes for these SPI devices manage the chip select pin, and require a pointer to a `GpioPinActiveLo` object for the chip select signal.

Notice that the MCU signal for the chip select pin of the non-volatile memory is actually the chip select signal of the SPI peripheral hardware. In this case, we have allocated a signal object of type class `GpioPinActiveLo`, so the pin is controlled through this signal object. If the object class was changed to `DigitalPin`, the pin would be under the control of the SPI peripheral.

In contrast, the MCU signal for the chip select of the temperature sensor is a general-purpose digital signal (GPIO). It can never be controlled by any peripheral hardware.

In general, a MCU signal that is allocated as a peripheral signal (non GPIO) can be used in two ways. If you define a signal object of type class `DigitalPin`, the pin multiplexer is set according to the selected MCU signal and the pin will be under the control of the peripheral. If you define a signal object of type class `GpioPin` (or an inherited class), the pin multiplexer is set to select the GPIO signal and the pin is controlled by software using that signal object.

Revision History

Revision	Date	Changes From Previous Release
1	22 Nov 2019	▪ Original release.