N to 1 Analog Multiplexer Datasheet AMuxN V 3.00

Copyright © 2009-2014 Cypress Semiconductor Corporation. All Rights Reserved.

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C28x45, CY8C28x43, CY8C28x13, CY8C28x52, CY8C28x33, CY8C21x45, CY8C22x45, CY8C24x94, CY7C64215						
Partial Mux Bus, Left (odd Port/Pins) or Partial Mux Bus, Right (even Port/Pins) Mux Bus	0	0	0	305**	5**	1 to N*
Single Mux Bus	0	0	0	279**	5**	1 to N*
Differential Mux Bus	0	0	0	285***	9***	1 to N*
CY8C20x34, CY8C21x34, CY8C21x34B, CY8C20xx6, CY8C20xx6A, CY8C20xx6AN, CY8C20xx6AS, CY8C20xx6H, CY8C20xx6L, CY8C20xx7S, CYRF89435, CY8C20065, CY8C24193, CY8C24493, CY7C69xxx	0	0	0	263*	5*	1 to N*
N* – varies from device to device ** – one input is assigned. *** – two Inputs are assigned.						

For one or more fully configured, functional example projects that use this user module go to www.cypress.com/psocexampleprojects.

Features and Overview

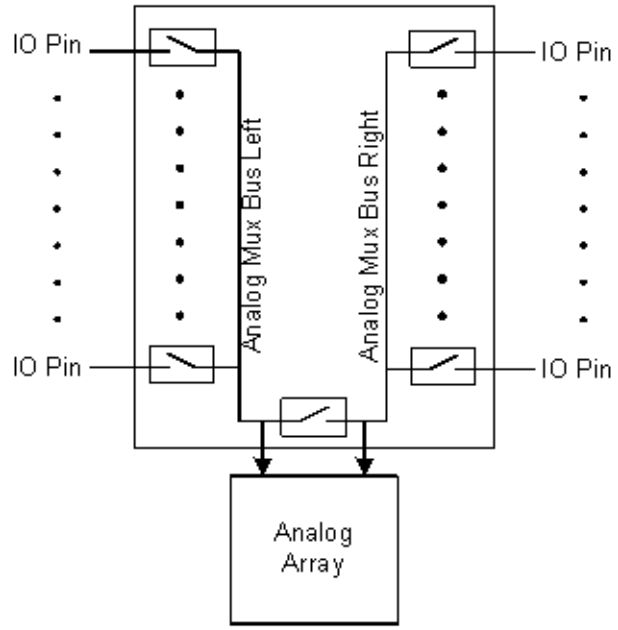
The AMuxN User Module provides APIs for multiplexing all (one or more at a time) GPIOs on the analog mux bus.

- High impedance input
- Input signals may be rail-to-rail
- Left Partial Mux Bus, Right Partial Mux Bus, Single Mux Bus, and Differential Mux Bus configurations
- API for multiplexing all GPIOs on the analog mux bus
- API for connecting or disconnecting AMux bus halves (legacy)

The AMuxN is used as an input multiplexer to an on-chip programmable gain amplifier (PGA) or an analog-to-digital converter (ADC); the static and signal voltage-dependent resistance of the mux is not a significant part of the system error budget. Because the mux is switched by software, the analog switch settling time is less compared to the time it takes to execute commands to switch the mux. The AMuxN

User Module is based on the analog mux bus hardware of the devices. This user module provides a simple API that enables you to easily connect any GPIO to the analog mux bus. This function allows many I/O pins to connect to a common internal analog bus. Use the AMuxN User Module when your application needs to dynamically choose from two or more ports during operation.

Figure 1. AMuxN Block Diagram

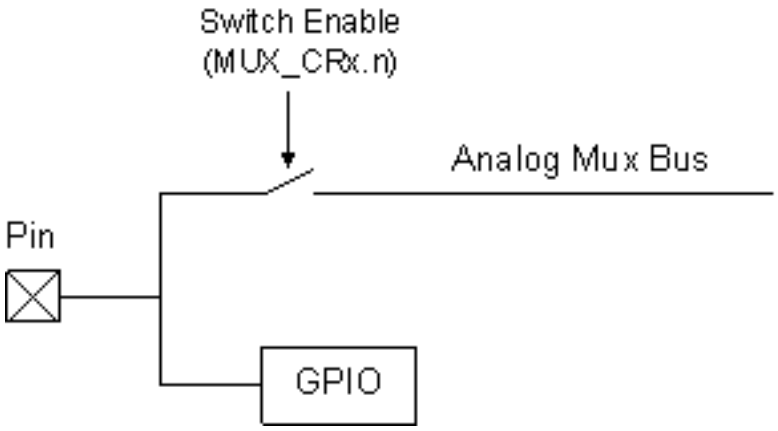


Functional Description

The AMuxN User Module provides an API to multiplex all of the GPIOs on the analog mux bus and to connect or disconnect the AMux bus halves. In the CY8C28x45, all I/O pins connect to this bus. Any number of pins can be connected simultaneously, and dedicated support circuitry allows selected pins to be alternately charged high or connected to the bus.

Pins are individually connected to the internal bus by setting the corresponding bits in the MUX_CRx registers. Any number of pins can be enabled at the same time. At reset, all of these mux connections are open (disconnected).

Figure 2. AMuxN Connection to Analog Mux Bus



Normally, the associated GPIO pin is put in a high impedance state for these applications, although there are cases where the GPIO cell is configured to briefly drive pin initialization states.

The analog bus can be connected as an input into either the positive or negative inputs of any analog continuous time (CT) block. The analog mux bus can be split into two separate nets. The two analog mux nets can be connected to different analog columns for simultaneous signal processing.

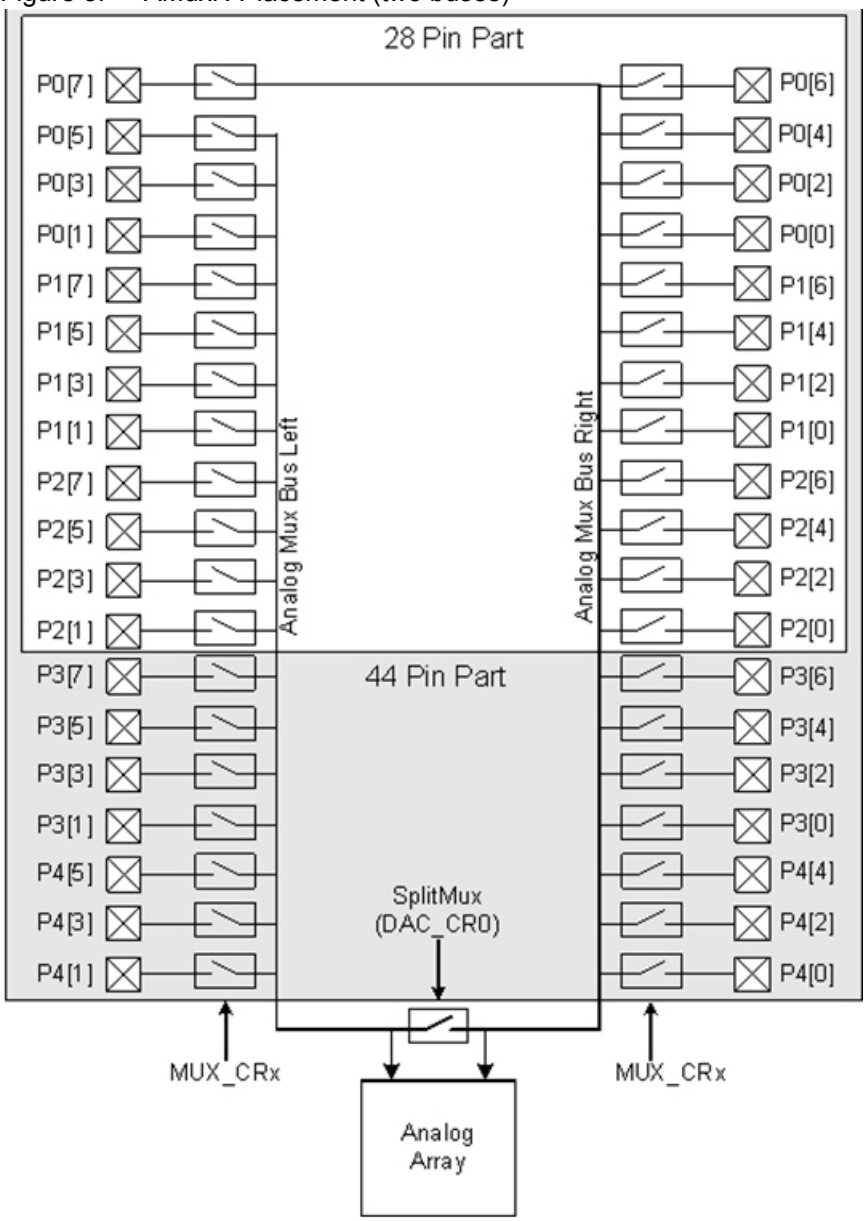
There are four user module configurations:

- Left Partial Mux Bus (odd port/pins)
- Right Partial Mux Bus (even port/pins)
- Single Mux Bus
- Differential Mux Bus, Left (odd port/pins), and Right (even port/pins)

The partial mux bus configuration only occupies one section of an analog mux bus. It only applies to PSoC devices that have multiple sections of analog mux bus. CY8C28x45 has two sections of analog mux bus. This configuration allows the use of only one section of the analog mux bus.

The AMuxN User Module maps into the analog mux bus as shown in Figure 3.

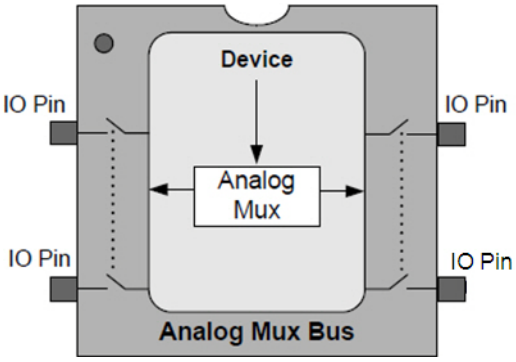
Figure 3. AMuxN Placement (two buses)



CY8C28x45 and CY8C22x45 can split the analog bus into two separate sections. In CY8C28x45, all GPIO pins are enabled for this connection. Odd pins are connected to one bus and even pins to the other bus. The two mux buses can be shorted together using the switch controlled by the SplitMux bit of the Analog Mux DAC Control Register (DAC_CR0).

The bidirectional nature of the analog mux switches allows a direct connection between any of the I/O pins. Enabling two (or more) pins at the same time connects these pins together, with approximately 400 ohms of resistance between each pin and the analog mux bus.

Figure 4. AMuxN Placement (one bus)



The analog global bus can be connected as a comparator input in the CY8C20xx6 devices.

DC and AC Electrical Characteristics

Unless otherwise specified in the following table, all limits are guaranteed for T_A = -40 °C to +85 °C, V_{DD} = 5.0 V ± 10%.

Table 1. 5.0 V AMuxN DC and AC Electrical Characteristics

Parameter	Conditions and Notes	Minimum	Typical	Limit	Units
Input leakage	Gross tested to 1 µA	–	1	–	nA
Input capacitance		0.5	1.7	8	pF
Bandwidth		–	10	–	MHz
Input voltage range		0	–	V_{DD}	V

Unless otherwise specified in this table, all limits are guaranteed for T_A = -40 °C to +85 °C, V_{DD} = 3.3 V ± 10%.

Table 2. 3.3 V AMuxN DC and AC Electrical Characteristics

Parameter	Conditions and Notes	Minimum	Typical	Limit	Units
Input leakage	Gross tested to 1 µA	–	†	–	nA
Input capacitance		0.5	1.7	8	pF
Bandwidth		–	10	–	MHz
Input voltage range		0	–	V_{DD}	V

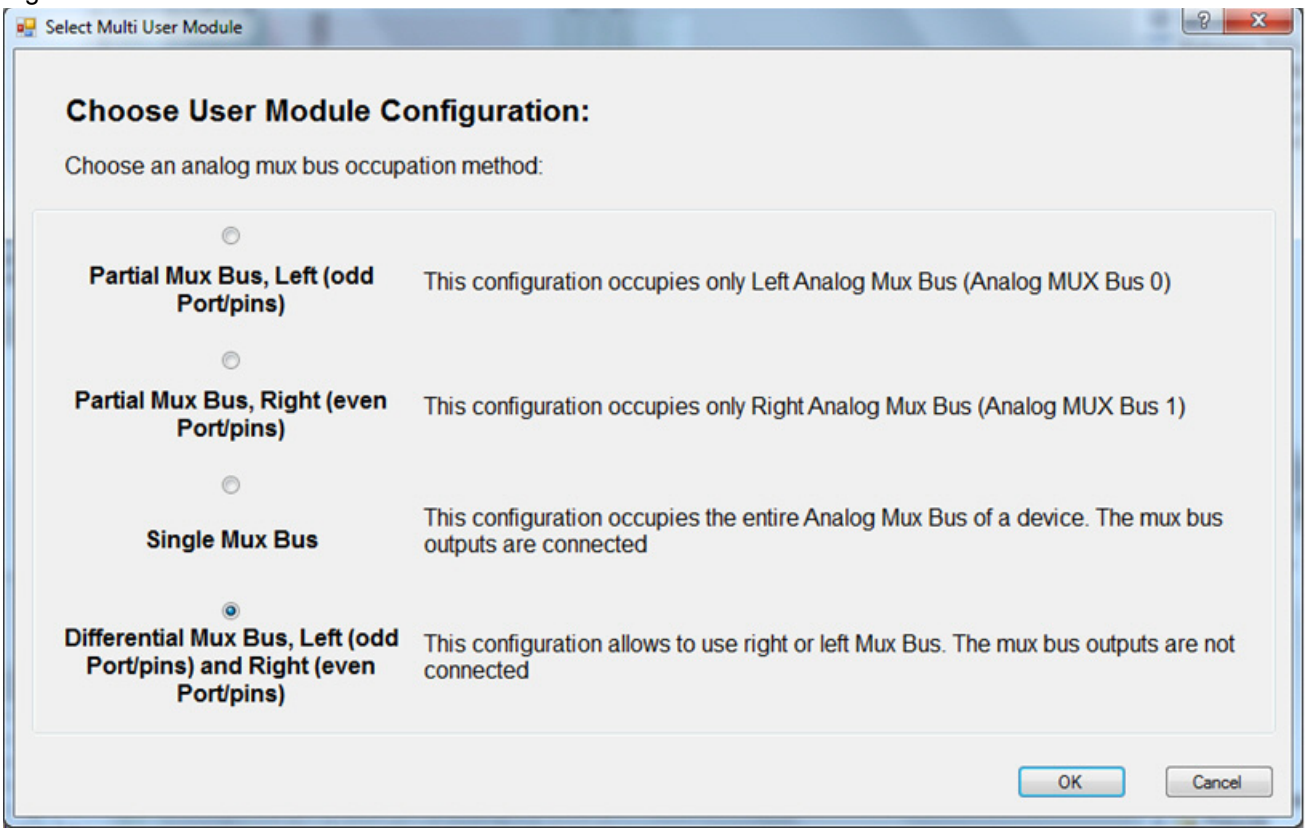
Note See the device datasheet for typical values.

Placement

AMuxN is a Multi User Module. The blocks for the user module are automatically placed when the user module is instantiated and user selections are entered. Here is a quick start guide for the placement and configuration of this user module.

1. Select and place the AMuxN User Module. The MUM wizard pops up and helps you to configure the user module. The configurations vary depending on the selected device.
2. For devices with two analog mux buses, such as the CY8C28x45, CY8C28x43, CY8C28x52, CY8C28x13, CY8C28x33, CY8C22x45, and CY8C21x45, a MUM wizard appears. Select one of the following configurations: Partial Mux Bus, Single Mux Bus, or Differential Mux Bus, and click **OK**.

Figure 5. AMUXN Multi User Module Wizard



3. For devices limited to a single mux such as CY8C24x94, CY8C20x34, CY8C21x34,CY8C20x34, there is no MUM Wizard and the default enabled option is Single Mux Bus.
4. Right-click the AMuxN User Module in Workspace Explorer or right-click the AMuxN User Module in the interconnect view as shown in the figure to access the AMuxN Wizard.

Wizard Pin Legend

White – The pins cannot be used as a AMux input

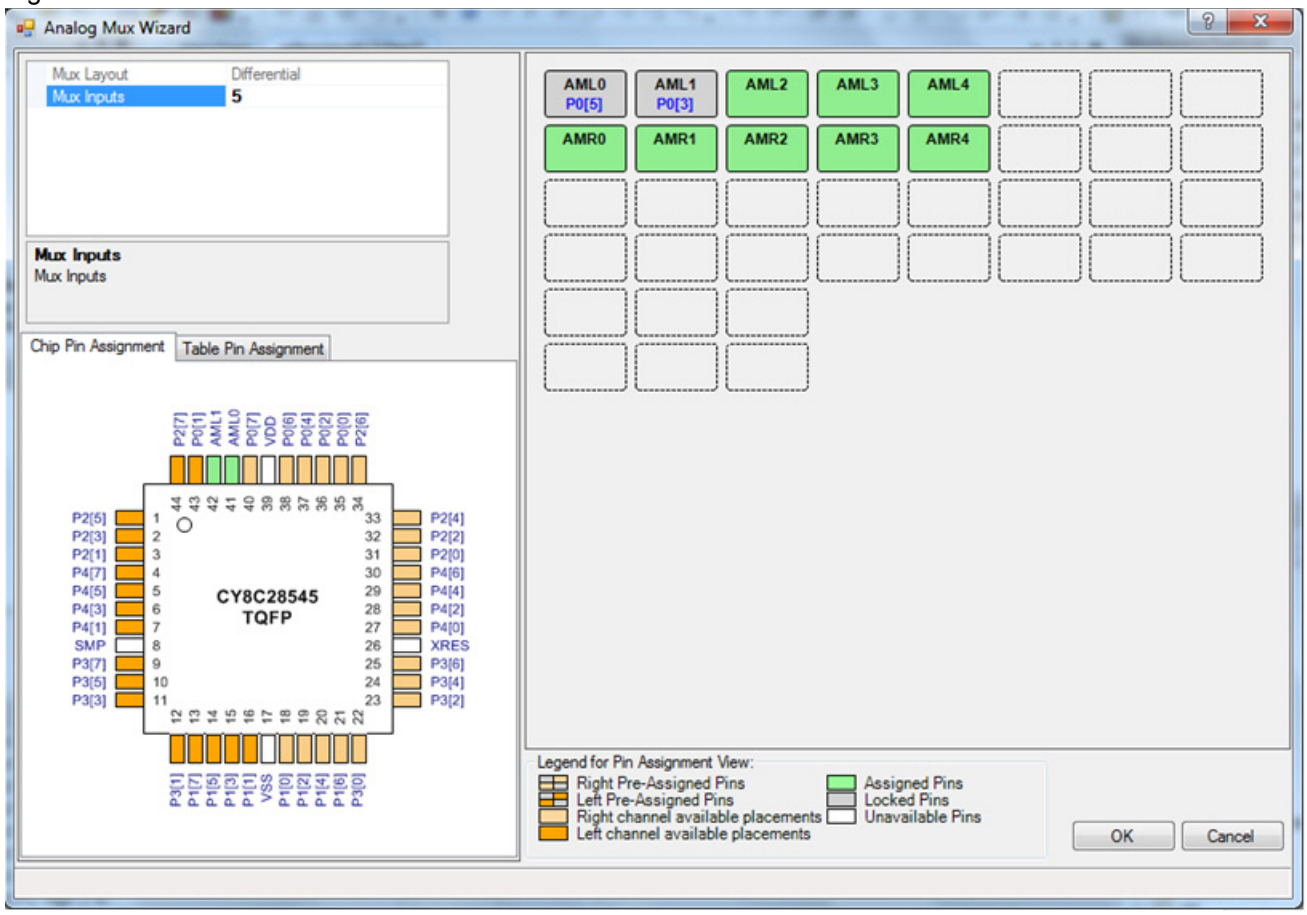
Gray – The pin is locked. There are two possible causes for this. The first possibility is that another user module such as the LCD or I^2C has claimed the pin. The second possibility is that the name of the pin has been changed from its default. To return the pin name to its

default, in the Pinout view expand the pin, and choose **Default** from the **Select** menu. The pin is now available for assignment in the wizard.

Orange – The pin is available for assignment.

Green – The pin is assigned as an AMux.

Figure 6. AMUXN User Module Wizard



5. Left-click an AMux pin and drag it to any available pin. The port pin is green after selection and it is no longer available. Change the AMux assignments by dragging the sensor of the port pin.
6. Repeat this procedure for the remaining AMux pins.
7. In the table assignment view, the pins are shown in the form of a table.
8. The Clear All Pins option is available by right-clicking the wizard ‘Chip Pin Assignment View’ and ‘Table Pin Assignment View’. This option unassigns all chip pins.
9. Click **OK** when the assignment is complete.

Parameters and Resources

There are no separate user-programmable parameters. All conditions and input selections are made through the wizard.

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow you to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the “include" files.

Note

In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X before the call if those values are required after the call. This “registers are volatile" policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must ensure their code observes the policy, too. Though some user module API functions may leave A and X unchanged, there is no guarantee they may do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

Global Variables

API functions use different global arrays. Do not alter these arrays manually. However, you can inspect these values for debugging purposes. For example, you can use a charting tool to display the contents of the arrays. There are two global arrays:

- AMuxN_bPinMask[]
- AMuxN_bPortMask[]
- **AMuxN_bPinMaskGlobal[]** - This is a BYTE array that contains the bit masks for each pin. The array size is equal to the pin count. The AMuxN_bPinMaskGlobal[] data is updated by the AMuxN_Start API.
- **AMuxN_bPort[]** - This is a BYTE array that contains the register address for the port number. The array size is equal to the pin count. The AMuxN_bPort[] data is updated by the AMuxN_Start API.
- **AMuxN_bPortMask[]** - This is a BYTE array, which contains the inverted pin masks for each used port. Number of values equals to TotalPortsCount. The AMuxN_bPortMask[] data is updated by the AMuxN_Start API.

Functions

The API functions are provided as part of the user module to allow you to deal with the module at a higher level.

AMuxN_Start

Description:

Initializes global variable AMuxN_bPort[] and AMuxN_bPinMaskGlobal[] arrays from AMuxN_Port and AMuxN_Pin_Table to execute the AMuxN_Select, AMuxN_Add and AMuxN_DeSelect functions more efficiently.

C Prototype:

```
void AMuxN_Start(void)
```

Assembly:

```
call  AMuxN_Start
```

Parameters:

None

Return Value:

None

Side Effects:

None

AMuxN_Select

Description:

This API causes the user-selected mux input to be the only analog input connection to the bus. The user's code passes a single value representing the mux input number. Calculation of the port and pin assignment from the mux input number is implemented in a table or a similar structure. For differential mux, the common mux number is used to identify inputs on right and left sides.

C Prototype:

```
void AMuxN_Select(BYTE bMuxNumber)
```

Assembly:

```
mov A, bMuxNumber
call  AMuxN_Select
```

Parameters:

bMuxNumber: Number of mux switches selected, 0 through (maximum value set by parameter)-1. Symbolic names are given in C and assembly, and their associated values are listed in the following table (Left Partial Mux Bus, Right Partial Mux Bus, and Single Mux Bus Configurations):

Symbolic Name	Value
AMuxN_INPUT_AM0	0
AMuxN_INPUT_AM1	1
AMuxN_INPUT_AM2	2
AMuxN_INPUT_AM3	3
AMuxN_INPUT_AMN_1	TotalInputsCount –1
TotalInputsCount – total value of assigned inputs	

Symbolic names for Differential Mux Bus Configuration:

Constant	Value
AMuxN_INPUT_AML0	0
AMuxN_INPUT_AMR0	1
AMuxN_INPUT_AML1	2
AMuxN_INPUT_AMR1	3
…	
AMuxN_INPUT_ AMLN-2	TotalInputsCount -2
AMuxN_INPUT_ AMRN-1	TotalInputsCount -1
TotalInputsCount – Total value of assigned inputs	

Return Value

None

AMuxN_Add

Description:

Connects a second (or other additional) mux input pin to the mux bus. This enables parallel connections for reduced resistance, enables a de-multiplexer when a static connection is connected to an analog output, and enables a cross-point switch.

C Prototype:

```
void  AMuxN_Add(BYTE bMuxNumber)
```

Assembly:

```
mov A, bMuxNumber
call  AMuxN_Add
```

Parameters:

bMuxNumber: Number of mux switches selected, 0 through (maximum value set by parameter) –1. Symbolic names are given in C and assembly, and their associated values are listed in the following table (Left Partial Mux Bus, Right Partial Mux Bus, and Single Mux Bus Configurations):

Symbolic Name	Value
AMuxN_INPUT_AM0	0
AMuxN_INPUT_AM1	1
AMuxN_INPUT_AM2	2
AMuxN_INPUT_AM3	3
AMuxN_INPUT_AMN_1	TotalInputsCount-1
TotalInputsCount – Total value of assigned inputs	

Symbolic names for Differential Mux Bus configuration:

Constant	Value
AMuxN_INPUT_AML0	0
AMuxN_INPUT_AMR0	1
AMuxN_INPUT_AML1	2
AMuxN_INPUT_AMR1	3
…	
AMuxN_INPUT_ AMLN-2	TotalInputsCount -2
AMuxN_INPUT_ AMRN-1	TotalInputsCount -1
TotalInputsCount – Total value of assigned inputs	

Return Value:

None

AMuxN_DeSelect

Description:

Disconnects selected mux input from the mux bus. The user's code shall pass a single value representing the mux input number. Calculation of the port and pin assignment from the mux input number shall be implemented in a table or similar structure. For differential mux, the common mux number is used to identify inputs on right and left sides, and both sides shall be deselected.

C Prototype:

```
void  AMuxN_DeSelect(BYTE bMuxNumber)
```

Assembly:

```
mov A, bMuxNumber
call  AMuxN_DeSelect
```

Parameters:

bMuxNumber: Number of mux switches selected, 0 through (maximum value set by parameter) -1. Symbolic names are given in C and assembly, and their associated values are listed in the following table (Left Partial Mux Bus, Right Partial Mux Bus and Single Mux Bus configurations):

Symbolic Name	Value
AMuxN_INPUT_AM0	0
AMuxN_INPUT_AM1	1

Symbolic Name	Value
AMuxN_INPUT_AM2	2
AMuxN_INPUT_AM3	3
AMuxN_INPUT_AMN_1	TotalInputsCount -1
TotalInputsCount – Total value of assigned inputs	

Symbolic names for Differential Mux Bus configuration:

Constant	Value
AMuxN_INPUT_AML0	0
AMuxN_INPUT_AMR0	1
AMuxN_INPUT_AML1	2
AMuxN_INPUT_AMR1	3
…	
AMuxN_INPUT_ AMLN-2	TotalInputsCount -2
AMuxN_INPUT_ AMRN-1	TotalInputsCount -1
TotalInputsCount – Total value of assigned inputs	

Return Value:

None

AMuxN_ClearAll

Description:

Disconnects all the mux inputs from the mux bus. For differential mux, disconnects all mux inputs from both mux buses.

C Prototype:

```
void  AMuxN_ClearAll(void)
```

Assembly:

```
lcall  AMuxN_ClearAll
```

Parameters:

None

Return Value:

None

Side Effects:

None

AMuxN_Stop

Description:

This function is only provided for API compatibility and performs no action.

C Prototype:

```
void  AMuxN_Stop(void)
```

Assembly:

```
lcall  AMuxN_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

None

Sample Firmware Source Code

Example 1:

```
;;-------------------------------------------------------------------
;; Sample ASM Code for the AMuxN user module
;;
;;-------------------------------------------------------------------

export _main

include "m8c.inc"
include "AMuxN.inc"

_main:

call   AMuxN_Start
mov  A, AMuxN_INPUT_AM0    ; specify number of mux switches selected
call   AMuxN_Select     ; Switch only Input0 to the Analog Mux Bus
mov  A, AMuxN_INPUT_AM1    ; specify number of mux switches selected
call   AMuxN_Add   ; Switch additional Input to the Analog Mux Bus

; …Other code
ret
```

The sample project written in C:

```
//-------------------------------------------------------------------
// Sample C Code for the AMuxN user module
//
//-------------------------------------------------------------------


#include "m8c.h"
```

```
#include "AMuxN.h"

void main(void)
{
    BYTE bMuxNumber;
    bMuxNumber = AMuxN_INPUT_AM0;    // Number of mux switches selected
    AMuxN_Start();
    AMuxN_Select(bMuxNumber);       // Switch only Input0 to the Analog Mux Bus
    bMuxNumber = AMuxN_INPUT_AM1;    // Number of mux switches selected
    AMuxN_Add(bMuxNumber);          // Switch additional Input to the Analog Mux Bus

    // …Other code
}
```

Configuration Registers

This register is configured by the initialization and API library. You do not have to change or read this register directly. This section is given as a reference.

Table 3. Register MUX_CRx

Bit	7	6	5	4	3	2	1	0
Value	ENABLE[7:0]							

The Analog Mux Port Bit Enables Registers (MUX_CRx) are used to control the connection between the analog mux bus and the corresponding pin. They are set by the AMuxN API.

Table 4. Register DAC_CR0

Bit	7	6	5	4	3	2	1	0
Value	SplitMux	NA	NA	NA	NA	NA	NA	NA

The SplitMux bit of Analog Mux DAC Control Registers (DAC_CRx) selects the split configuration for the CY8C28x45. It is set by the selecting the Left Partial Mux Bus or Right Partial Mux Bus configuration and reset by the selecting the Whole Mux Bus configuration.

Version History

Version	Originator	Description
1.0	DHA	Initial Version
1.0.b	DHA	Added wizard help file.
1.0.c	DHA	Updated the descriptions of the Left and Right MUM configurations by adding references to Analog Mux Bus 0 and 1.
2.00	DHA	1. Added wizard and new API to user module. 2. Extended support for other PSoC families. 3. Added symbolic Input Names in user module header files. 4. Updated ROM and RAM usage in the user module datasheet.
3.00	HPHA	1. Fixed problem handling pins in the User Module wizard when the configuration is changed. 2. Added new User Module configurations to support CY8C24x94 (backward incompatible) and CY7C64215 families.
3.00b	HPHA	Added build errors if AMuxN UM Wizard did not run.

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.

Copyright © 2009-2014 Cypress Semiconductor Corporation. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer™ and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.