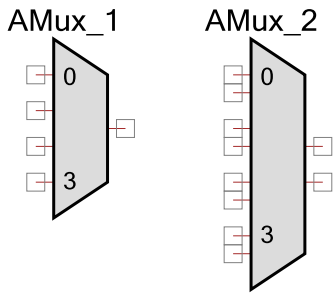
Analog Multiplexer (AMux)
1.20

Features



- Single or differential inputs
- Adjustable between 2 and 32 inputs
- Software controlled
- Inputs may be pins or internal sources
- Multiple simultaneous connections
- Bi-directional (passive)

General Description

The analog multiplexer (AMux) component can be used to connect none, one, or more analog signals to a different common analog signal. The ability to connect more than one analog signal at a time provides cross-bar switch support, which is an extension beyond traditional mux functionality.

When to use an AMux

The AMux should be used any time multiple analog signals must be multiplexed into a single source or destination. Since the AMux is passive, it can be used to multiplex input or output signals.

Input/Output Connections

This section describes the various input and output connections for the AMux. An asterisk (*) in the list of I/O's states that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

aN – Analog

The AMux is capable of having between 2 and 32 analog inputs.

bN – Analog *

The paired inputs (aN,bN) are only used when the MuxType parameter is set to "Differential."

y – Analog (Required)

The “y” signal is the common connection. Whichever aN signal is selected with the Select function will be connected to this terminal.

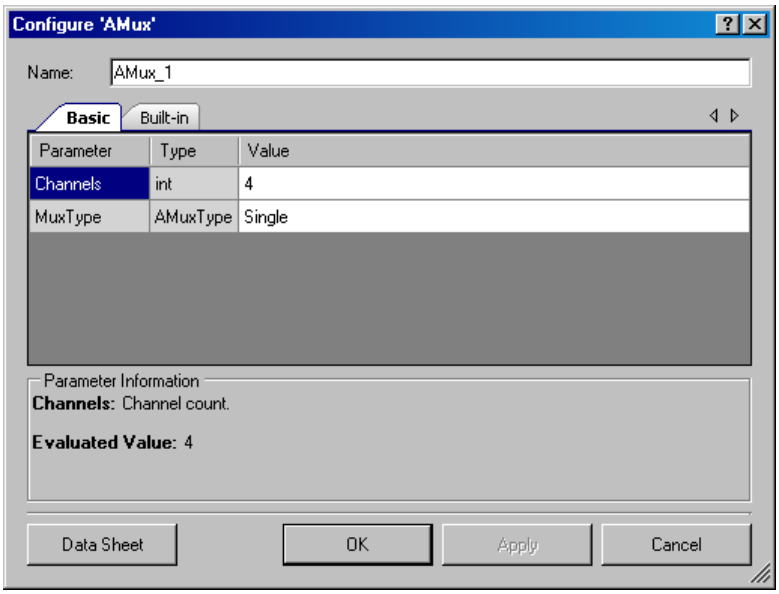
x – Analog *

The “x” signal is the common connection for the second input, bN, when using a differential mux. Whichever signal is selected with the Select() function will be connected to this terminal.

Parameters and Setup

Drag an AMux component onto your design and double-click it to open the Configure dialog.

Figure 1 Configure AMux Dialog



The AMux provides the following parameters.

Channels

This parameter selects the number of inputs or paired inputs depending on the MuxType. Any value between 2 and 32 may be selected.

MuxType

This parameter selects between a single input per connection “Single” or a dual input “Differential” input mux. A type “Single” is used when the input signals are all referenced to the same signal such as Vssa. In cases where two or more signals may have different signal

reference, the “Differential” option should be selected. The differential mode is most often used with an ADC that provides a differential input.

Placement

There are no placement specific options.

Resources

The AMux makes use of the individual switches that connect blocks to analog busses and analog busses that connect to pins.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "AMux_1" to the first instance of a component in a given design. You can the rename the instance to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "AMux".

Function	Description
void AMux_Start(void)	Resets all inputs
void AMux_Stop(void)	Resets all inputs
void AMux_Select(uint8 chan)	Disconnect all channels then connect “chan”
void AMux_Connect(uint8 chan)	Connect “chan” signal, but do not disconnect other channels.
void AMux_Disconnect(uint8 chan)	Disconnect only “chan” signal
void AMux_FastSelect(uint8 chan)	Disconnect the last channel that was selected by this function, then connect the new signal “chan”.
void AMux_DisconnectAll(void)	Disconnect all channels, same as Start()

void AMux_Start(void)

Description: Disconnects all channels.

Parameters: None

Return Value: None

Side Effects: None

void AMux_Stop(void)

Description: Disconnects all channels.

Parameters: None

Return Value: None

Side Effects: None

void AMux_Select(uint8 chan)

Description: The Select function first disconnects all other channels, then connects the selected “chan” signal.

Parameters: (uint8) chan: The channel to connect to the common or output terminal.

Return Value: None

Side Effects: Connections made either by Connect or FastSelect will be disconnected when using Select.

void AMux_FastSelect(uint8 chan)

Description: This function first disconnects the last connection made with FastSelect or Select functions, then connects the given channel. The FastSelect function is similar to the Select function, except it is faster since it only disconnects the last channel selected – not all possible channels.

Parameters: (uint8) chan: The channel to connect to the common or output terminal.

Return Value: None

Side Effects: If the Connect function was used to select a channel prior to calling FastSelect, the channel selected by Connect will not be disconnected. This is useful when parallel signals need to be connected.

void AMux_Connect(uint8 chan)

Description: This function connects the given channel to the common signal without affecting any previous channel connection.

Parameters: (uint8) chan: The channel to connect to the common or output terminal.

Return Value: None

Side Effects: Calling the function Select will disconnect any channel connected with the Connect function before connecting the channel passed to the Select command.

void AMux_Disconnect(chan)

Description: Disconnect only the specified channel from the common or output terminal.

Parameters: (uint8) chan: The channel to disconnect from the common or output terminal.

Return Value: None

Side Effects: None

void AMux_DisconnectAll(void)

Description: Disconnects all channels.

Parameters: None

Return Value: None

Side Effects: None

Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the AMux component. This example assumes the component has been placed in a design with the default name "AMux_1."

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

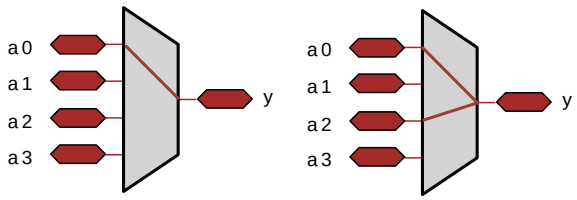
```
#include <device.h>

void main()
{
    AMux_1_Start( );          /* Reset all channels */
    AMux_1_Select(1);         /* Connect channel 1 */
}
```

Functional Description

The AMux is not like most hardware AMuxes. Two things make the AMux different from a standard fixed hardware mux: 1) It is a collection of independent switches, and 2) It is controlled by firmware not hardware.

Because of these two differences, this mux is flexible and allows more than one signal at a time to be connected to the common output signal. Two or more signals may be connected to the common output at any given time. Although in differential mode, the firmware will not allow the differential signals to connect to each other, it appears as two parallel muxes controlled by the same signal.

Figure 2 AMux may have multiple connections



DC and AC Electrical Characteristics

The AMux will operate at all valid supply voltages.

5.0V/3.3V DC and AC Electrical Characteristics

Parameter	Typical	Min	Max	Units	Conditions and Notes
Input					
Input Voltage Range	---		Vss to Vdd	V	
Capacitance to ground	---		---	pF	
Series resistance	---		---	Ω	

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.20.e	Minor datasheet edits.	
1.20.d	Minor datasheet edits.	
1.20.a	Added information to the component that advertizes its compatibility with silicon revisions.	The tool reports an error/warning if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
1.20	Symbol picture updated.	Updated to comply with corporate standard.

© Cypress Semiconductor Corporation, 2008-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.