8 to 1 Analog Multiplexer Datasheet AMux8 V 1.1

Copyright © 2001-2015 Cypress Semiconductor Corporation. All Rights Reserved.

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C29/27/24/22/23/21xxx, CY8CLED02/04/08/16, CY8CLED0xD, CY8CLED0xG, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CPLC20, CY8CLED16P01, CY8C28x43, CY8C28x13, CY8C28x52, CY7C64215						
	0	0	0	44	0	1 to 8

Features and Overview

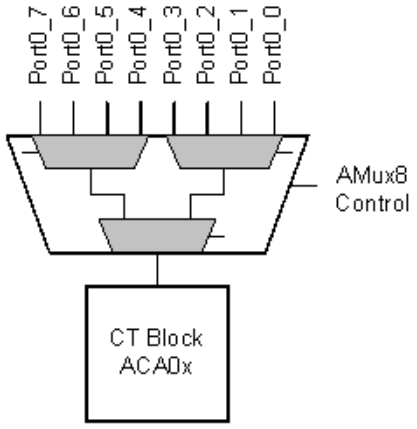
- High impedance input
- Input signals may be rail-to-rail
- Can be used with RefMux to multiplex input signals to switch capacitor block
- Programmable control of input source

The AMUX8 User Module provides an eight-input-analog-signal-multiplexer to a Continuous Time (CT) block, controlled by an API. One of four input signals may be selected to the input of the amplifier in the CT block. These input signals are connected to fixed ports, depending on which column the user module is placed. The module is also used in conjunction with a RefMux to route the multiplexed signals to the analog column bus.

The AMUX8 User Module is used when the application needs to dynamically select from two or more ports during operation.

If the input pin to the CT does not need to change during program operation, use the Device Editor to select the desired pin. This is done by left-clicking the AInMux_n mux.

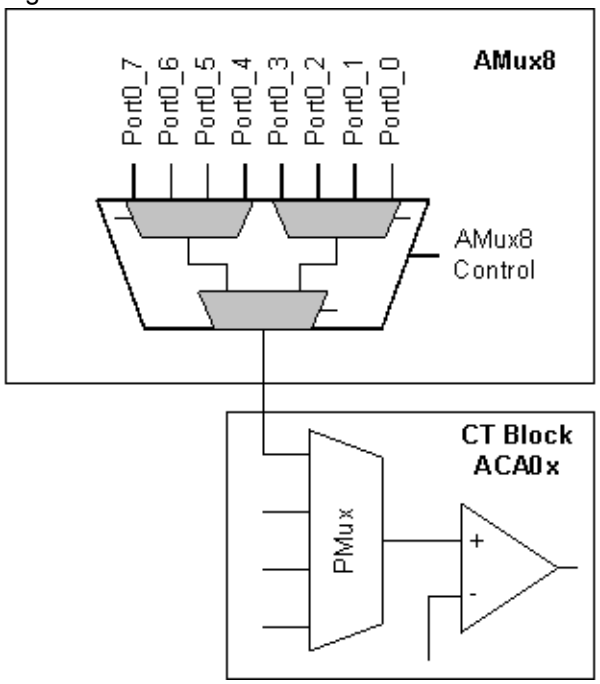
Figure 1. AMux8 Block Diagram



Functional Description

The AMUX8 User Module gives an API to control two analog input multiplexers (AInMux_x) and the analog columns 1 and 2. The output of the AInMux_n is routed to the PMux (Positive input Mux) of the amplifier in the CT block. The PMux in the CT block can be configured to accept the input from the AMUX8 User Module, by selecting the “AnalogInput_ColumnMux_n" for one of the inputs.

Figure 2. AMUX8 Connection to CT Block



DC and AC Electrical Characteristics

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

Table 1. 5.0V AMUX8 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 the following table, all limits guaranteed for T_A = –40°C to +85°C, V_{DD} = 3.3V ± 10%.

Table 2. 3.3V AMUX8 DC and AC Electrical Characteristics

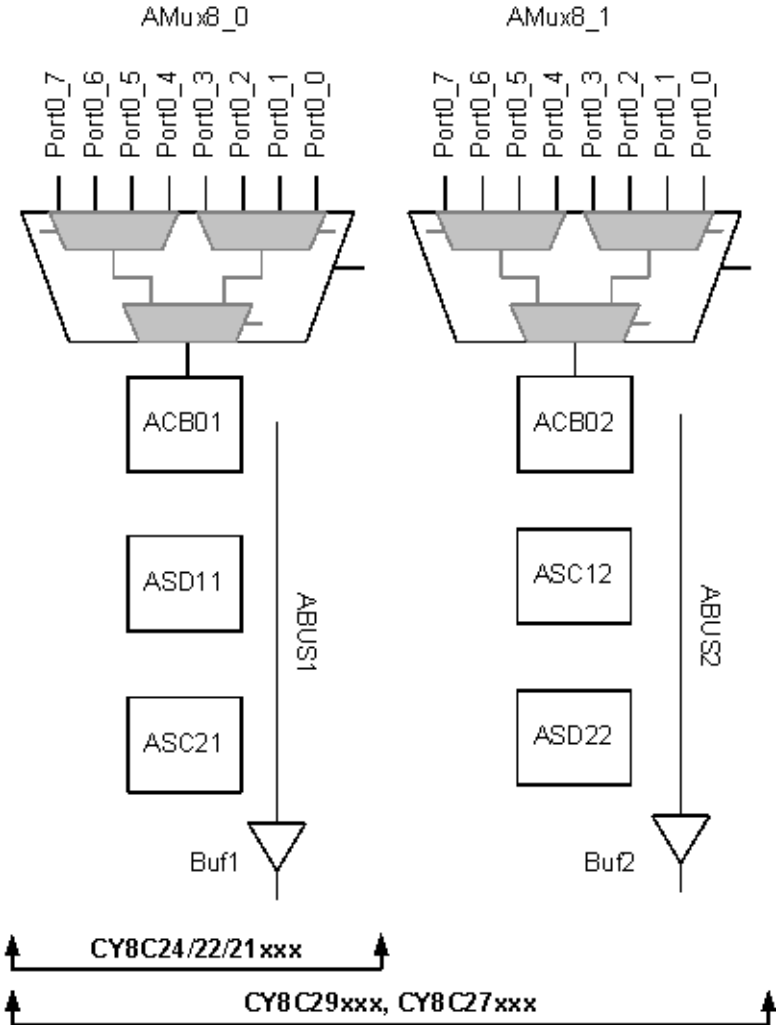
Parameter	Conditions and Notes	Minimum	Typical	Limit	Units
Input Leakage	Gross tested to 1 µA	--	†	--	nA
Input Capacitance		0.5	1.7	8	pF
Bandwidth		--	10	--	MHz
Input Voltage Range		0	--	V_{DD}	V

Note See the device datasheet for typical values.

Placement

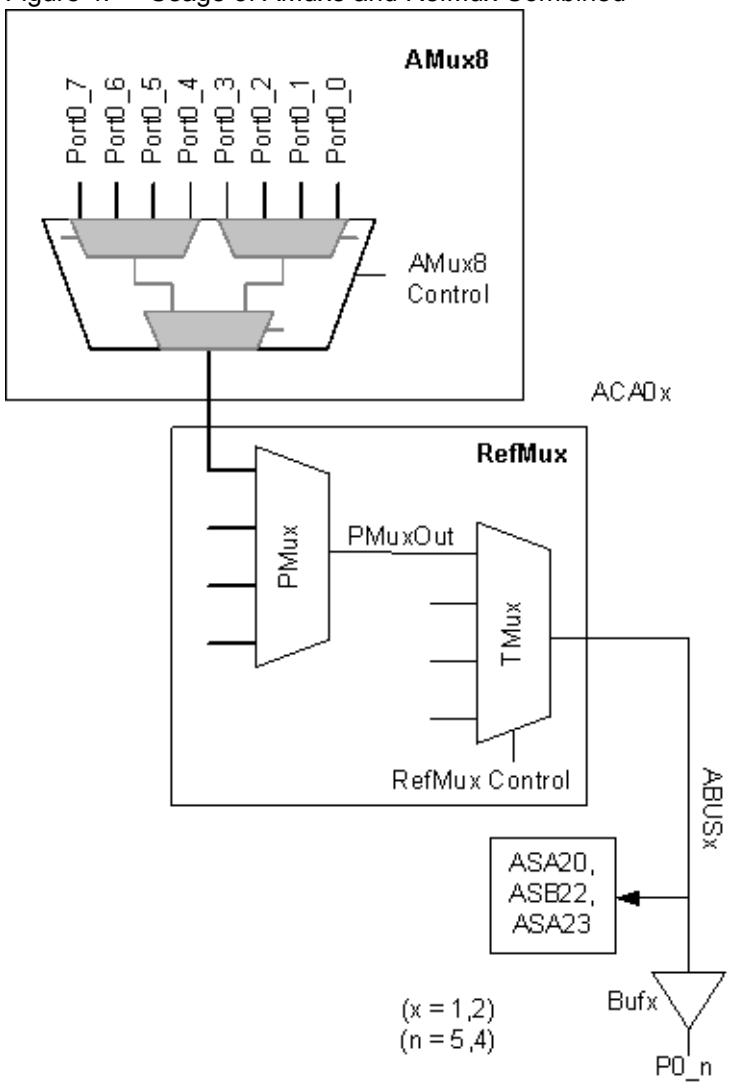
The AMUX8 User Module maps into any of the four AInMux blocks, as shown in Figure 3:

Figure 3. AMUX8 Placement



If used in conjunction with a RefMux User Module, the eight input signals may be multiplexed onto the analog column bus. This allows routing rail-to-rail signals into some of the switched capacitor blocks (or user modules) such as filters and ADCs. The AMUX8 User Module does not use a CT block, but connects to most user modules that do use CT blocks.

Figure 4. Usage of AMux8 and RefMux Combined



Parameters and Resources

Note Configurable parameters are not available for CY8C24X23 devices.

Analog Column Mux

This parameter selects the multiplexer. Valid inputs are 1 or 2 for analog columns 1 and 2. The column second from the left is ‘1‘and column to its right is ‘2‘. Use the multiplexer AInMux_n mux, directly above the CT block in the selected analog column.

AInMux

The AMUX8 User Module uses two of the four AInMux analog multiplexers resources to select one of eight input pins.

Application Programming Interface (API)

The Application Programming Interface (API) routines are given as part of the user module to allow the designer 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 function 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.

AMUX8_InputSelect

Description:

Switches selected port to the CT block.

C Prototype:

```
void  AMUX8_InputSelect(BYTE bChannel);
```

Assembly:

```
mov   A, AMUX8_PORT0_3
lcall  AMUX8_InputSelect
```

Parameters:

bChannel: One byte that specifies which port pin is connected to the CT Block. The usable ports are all input pins to port0, 0 thru 7. Symbolic names are given in C and assembly, and their associated values, are listed in the following table:

Symbolic Name	Value
AMUX8_PORT0_0	0x00
AMUX8_PORT0_1	0x01
AMUX8_PORT0_2	0x02
AMUX8_PORT0_3	0x03
AMUX8_PORT0_4	0x04
AMUX8_PORT0_5	0x05
AMUX8_PORT0_6	0x06
AMUX8_PORT0_7	0x07

Note The symbolic names are generated regardless of whether the device you are using has these ports available or not. Do not use the symbolic names associated with pins that do not exist on your device.

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

AMUX8_Start

Description:

At present, this function performs no action and is not required for operation of the AMUX8. It is given for compatibility only.

C Prototype:

```
void  AMUX8_Start(void);
```

Assembly:

```
lcall  AMUX8_Start
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

AMUX8_Stop

Description:

Currently this function performs no action and is not required for operation of the AMUX8 User Module. It is given for compatibility only.

C Prototype:

```
void  AMUX8_Stop(void);
```

Assembly:

```
lcall  AMUX8_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

Sample Firmware Source Code

Here is a simple assembly example for selecting an analog signal using AMUX8:

```
;;-------------------------------------------------------------------
;; Sample ASM Code for the AMUX8 user module
;;
;;-------------------------------------------------------------------

export _main

include "m8c.inc"
include "AMUX8.inc"

_main:

    mov  A,AMUX8_PORT0_3       ; specify port pin Port0_3
    call AMUX8_InputSelect     ; to connect to the CT block

    ; …Other code
    ret
```

The sample project written in C is:

```
//-------------------------------------------------------------------
// Sample C Code for the AMUX8 user module
//
//-------------------------------------------------------------------


#include "m8c.h"
#include "AMUX8.h"

void main(void)
{
   BYTE bPortNumber;

bPortNumber = AMUX8_PORT0_3;          // Assign port number
   AMUX8_InputSelect(bPortNumber);       // Switch Port0, pin3 to the
                                      // CT block.


   // …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 AMX_IN

Bit	7	6	5	4	3	2	1	0
Value	ACI3[1]	ACI3[0]	ACI2[1]	ACI2[0]	ACI1[1]	ACI1[0]	ACI0[1]	ACI0[0]

ACI3[1:0]: Mux control bits for AInMux_3. ACI2[1:0]: Mux control bits for AInMux_2. ACI1[1:0]: Mux control bits for AInMux_1. ACI0[1:0]: Mux control bits for AInMux_0.

Table 4. Register ABF_CR0

Bit	7	6	5	4	3	2	1	0
Value	ACol1Mux	ACol2Mux	na	na	na	na	na	na

Acol1Mux switches between Analog Column muxes 0 and 1. ACol2Mux switches between Analog Column muxes 2 and 3.

Version History

Version	Originator	Description
1.1	DHA	Added Version History

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 © 2001-2015 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.