DMX512 Receiver Datasheet DMX512Rx V 1.0

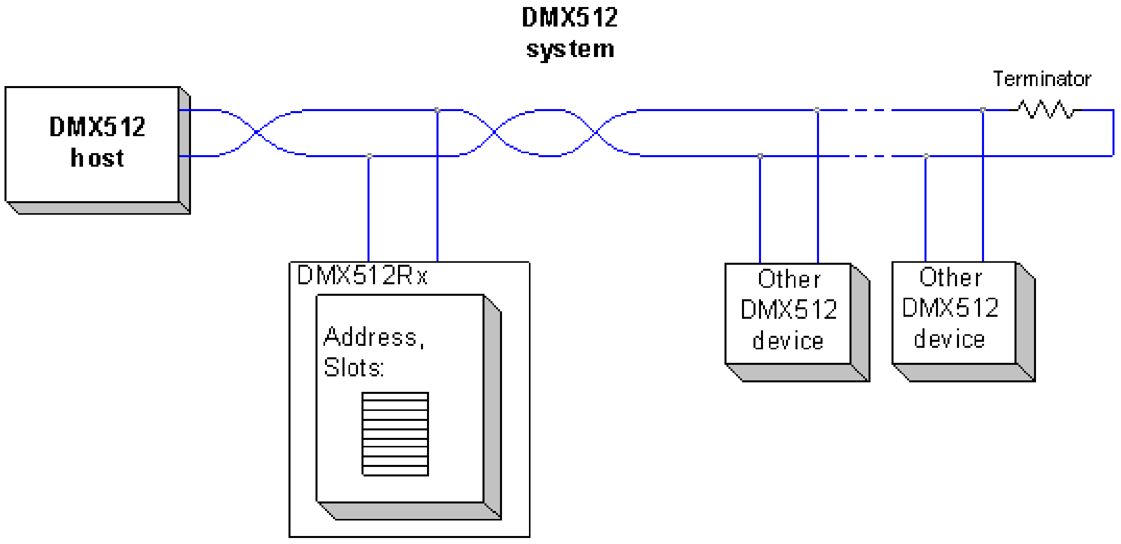
Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8CLED02/04/08/16, CY8CLED0xD, CY8CLED0xG, CY8CLED16P01, CY8C29x66, CY8C27x43, CY8C24x94, CY8C21x23	2	0	0	242	11	1

Features and Overview

- 250 kbps DMX512 protocol receiver
- Selectable address
- Selectable captured slots count
- Interrupt on byte received
- Interrupt on start-of-frame

The DMX512Rx User Module is used to receive data via the DMX512 bus and store it in RAM, similar to the EzI2Cs User Module. The user module is composed of two digital blocks.

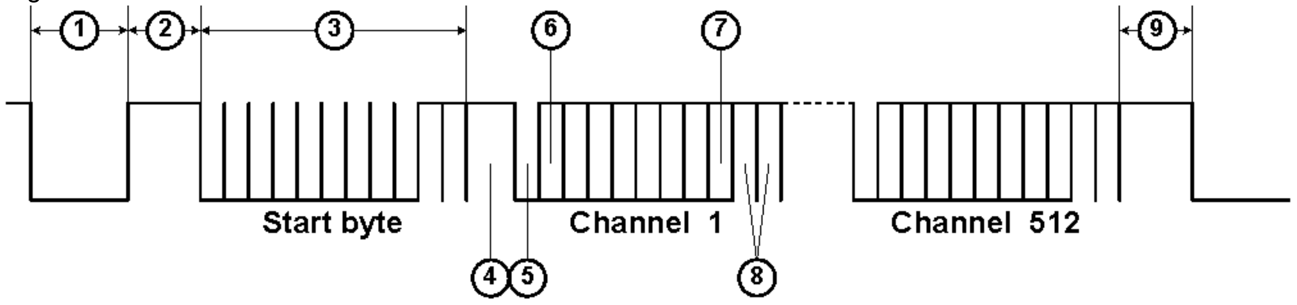
Figure 1. DMX512Rx Block Diagram



Functional Description

The DMX512 data stream is passed in the form of a packet that is repeated continually. This data packet consists of the synchro preamble which informs the receiver of the start of a packet. The preamble is followed by a serial data frame. The data frame contains the values for each channel in use. The minimum channel count is 24. The maximum is 512. The duration of every bit for DMX512 protocol is 4 µs. If the information is given over all 512 channels, then the maximum possible frequency of information update is 44 Hz.

Figure 2. DMX512 Data Packet



Reference	Description	Duration
1	Space for break (reset)	min 88 µs
2	Mark after break (MAB)	8 µs - 1s
3	Slot time	44 µs
4	Mark time between slots	0 µs -1s
5	Start bit	4 µs
6	Least significant data bit	4 µs
7	Most significant data bit	4 µs
8	Stop bit	4 µs
9	Mark before break (MBB)	0 µs -1s

The DMX512Rx User Module employs two digital PSoC blocks:

- One RX block for the data receiver
- One Pulse Width Discriminator (PWD) for detecting the break signal

The configuration of the underlying connective hardware, the digital PSoC blocks, coordinates the operation of the PSoC blocks as a single DMX512Rx User Module.

The RX and PWD operate independently. Each have their own registers and interrupts. They share the same enable and input. Setting the enable bit in the RX and PWD control registers enables the DMX512Rx User Module for operation. Use the provided API functions to enable and disable the user module.

The DMX512Rx User Module uses different clocks for RX and PWD components. The clock frequency for the RX must be exactly 2 MHz to meet 4 µs data bits requirement. The clock frequency for the PWD must be between 135 kHz and 166 kHz so the PWD block can catch the space for break.

The RX block uses the UART RX function of the communication digital block with no parity settings. The PWD block uses CRCPRS function of the digital block with the polynomial register equal to zero and the compare register set to 255.

The DMX512Rx User Module provides one interrupt each time space for break is detected and one interrupt for each byte received by the RX block. When data is sent with the mark time between slots set to zero, the interrupt occurrence frequency is equal to 44 µs. If the CPU clock is set to 24 MHz, the time required to execute the interrupt service routine (ISR) for bytes that are included into the active slot is 5.7 µs. For this reason, a CPU clock of 12 MHz is recommended. In no case should the CPU clock be set less than 6 MHz.

DC and AC Electrical Characteristics

Table 1. DMX512Rx AC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
RX input frequency	--	2	MHz	
PWD input frequency	--	135...166	kHz	
Allowed CPU clock range	--	6...24	MHz	

Placement

The RX block of the user module may be placed in any digital communication block and the PWD block of the user module may be placed in any digital block, but both blocks should be placed in the same digital row. The same input source is used for both RX and PWD components.

Parameters and Resources

RX Clock

Clock source for the RX block. Choose any one of the available sources. The Global I/O buses may be used to connect the clock input to an external pin or a clock function generated by a different PSoC block. When using an external digital clock for the block, the row input synchronization should be turned off for best accuracy, and sleep operation. One of the divided clocks (VC1, VC2, or VC3), or another PSoC block output can be specified as the clock input. The clock for RX block must be equal to 2 MHz for correct user module operations.

PWD Clock

Clock source for the PWD block. Choose any one of the available sources. The Global I/O buses may be used to connect the clock input to an external pin or a clock function generated by a different PSoC block. When using an external digital clock for the block, the row input synchronization should be turned off for best accuracy, and sleep operation. One of the divided clocks (VC1, VC2, or VC3), or another PSoC block output can be specified as the clock input. The clock for PWD block should be set in the range of 135 to 166 kHz for correct user module operations.

Input

The input can be connected to a low, a high, the analog comparator output bus, or one of the global buses. Using the global bus, the input can be connected to one of the external pins.

InvertInput

This parameter gives you the ability to invert the InputDataStream.

StartSlotID

The StartSlotID parameter contains a value ranging from 1 to 512. This parameter defines first slot ID that will be captured.

ClockSync

In the PSoC devices, digital blocks may provide clock sources in addition to the system clocks. Digital clock sources may even be chained in ripple fashion. This introduces skew with respect to the system clocks. This parameter may be used to control clock skew and ensure proper operation when reading and writing PSoC block register values. Appropriate values for this parameter should be determined from the following table.

ClockSync Value	Use
Sync to SysClk	Use this setting for any 24 MHz (SysClk) derived clock source that is divided by two or more. Examples include VC1, VC2, VC3 (when VC3 is driven by SysClk), 32KHz, and digital PSoC blocks with SysClk-based sources. Externally generated clock sources should also use this value to ensure that proper synchronization occurs.
Sync to SysClk*2	Use this setting for any 48 MHz (SysClk*2) based clock unless the resulting frequency is 48 MHz (in other words, when the product of all divisors is 1).
Unsynchronized	Use when unsynchronized inputs are desired.

Application Programming Interface

The Application Programming Interface (API) routines are provided 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 prior to 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 will 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.

There are nine functions in this API:

- DMX512Rx_Start
- DMX512Rx_Stop
- DMX512Rx_SetStartSlotID
- DMX512Rx_SetRamBuffer
- DMX512Rx_bGetBusActivity

- DMX512Rx_bGetSlotActivity
- DMX512Rx_bGetStartCode
- DMX512Rx_EnableInt
- DMX512Rx_DisableInt

DMX512Rx_Start

Description:

Enables the DMX512Rx User Module for operation.

C Prototype:

```
void  DMX512Rx_Start(void)
```

Assembler:

```
lcall  DMX512Rx_Start
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

DMX512Rx_Stop

Description:

Disables the DMX512Rx User Module.

C Prototype:

```
void  DMX512Rx_Stop(void)
```

Assembler:

```
lcall  DMX512Rx_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

DMX512Rx_SetStartSlotID

Description:

Redefines first slot ID that will be captured. Call this function only after the DMX512Rx_Start() function, because the Start function sets the first slot ID to the value entered in the Device Editor parameters. This function is optional and only needed if you change the first slot ID from the value set in the device editor. New StartSlotID value will be applied as soon as Space For Break symbol is detected in input bitstream. This guarantees that User Buffer will remain in consistent state even when StartSlotID is altered during data reception.

C Prototype:

```
void  DMX512Rx_SetStartSlotID(WORD wStartSlotID)
```

Assembler:

```
mov   A, <[wStartSlotID + 1]
mov   X, >[wStartSlotID + 0]
lcall  DMX512Rx_SetStartSlotID
```

Parameters:

wStartSlotID: A value from 1 to 512. The parameter value is passed in the A and X registers.

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

DMX512Rx_SetRamBuffer

Description:

This function sets the location and size of the RAM buffer (up to 256 bytes) that is used by DMX512 slots. This function is required and must be called before calling DMX512Rx_Start(). This function can also be called at any time after calling DMX512Rx_Start() but if it is called during data reception current receive operation will be immediatelly aborted to prevent memory corruption. Previous user buffer will possibly remain in inconsistent state with partially received packet. Reception will be resumed as soon as Space For Break symbol is detected in input bitstream.

C Prototype:

```
void  DMX512Rx_SetRamBuffer(BYTE bSize, (BYTE *)pAddr)
```

Assembler:

```
mov   A, >pAddr
push  A
mov   A, <pAddr
push  A
mov   A, [bSize]
push  A
lcall  DMX512Rx_SetRamBufer
add   SP,-3
```

Parameters:

bSize: Size of the data structure available to DMX512 slots.

pAddr: A pointer to the data array

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

DMX512Rx_bGetBusActivity

Description:

This function returns a nonzero value if a DMX512 bus activity occurred since last time this function was called. The activity flag is reset to zero before the function returns.

C Prototype:

```
BYTE  DMX512Rx_bGetBusActivity()
```

Assembler:

```
lcall  DMX512Rx_bGetBusActivity
mov    [bActivityFlag],A
```

Parameters:

None

Return Value:

This function returns zero value if no DMX512 bus activity is detected since the last time the function has been called. Otherwise it returns one of the following values :

Return Value	HEX Value	Meaning
DMX512Rx_SFB_RECEIVED	0x01	Space For Break sequence was detected in input bitstream.
DMX512Rx_SC_RECEIVED	0x02	StartCode was received. Use DMX512Rx_bGetStartCode function to retrive Start Code value.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

DMX512Rx_bGetSlotActivity

Description:

This function returns a non-zero value if a read of last required DMX512 slot occurred since last time this function was called. The activity flag resets to zero at the end of this function call.

C Prototype:

```
BYTE  DMX512Rx_bGetSlotActivity()
```

Assembler:

```
lcall  DMX512Rx_bGetSlotActivity
mov    [bActivityFlag],A
```

Parameters:

None

Return Value:

Returns a nonzero value if a read of the last required DMX512 slot has occurred. Returns zero otherwise.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

DMX512Rx_bGetStartCode

Description:

This function returns Start Code value retrieved from recently received DMX512 packet. Start Code is a data byte that precedes the DMX512 slots and tells the receiver what kind of data will follow. Return value is valid only when at least one Start Code has been received. Use DMX512Rx_bGetBusActivity function to detect reception of Start Code.

C Prototype:

```
BYTE  DMX512Rx_bGetStartCode()
```

Assembler:

```
lcall  DMX512Rx_bGetStartCode
mov    [bStartCode],A
```

Parameters:

None

Return Value:

BYTE bStartCode: Start Code retived from recently received DMX512 packet.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

DMX512Rx_EnableInt

Description:

Enables the interrupts for the DMX512Rx RX and PWD blocks.

C Prototype:

```
void  DMX512Rx_EnableInt(void)
```

Assembler:

```
lcall  DMX512Rx_EnabeInt
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

DMX512Rx_DisableInt

Description:

Disables the interrupts for the DMX512Rx RX and PWD blocks.

C Prototype:

```
void  DMX512Rx_DisableInt(void)
```

Assembler:

```
lcall  SSDM_DisableInt
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8CLED16). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

Sample Firmware Source Code

In the following examples, the correspondence between the C and assembly code is simple and direct.

The following is assembly language source that illustrates the use of the APIs.

```
include "PSoCAPI.inc"   ; PSoC API definitions for all user modules

area bss
DMX_RAM_BUF_SIZE: equ 8
DMX_RAM_BUF: blk DMX_RAM_BUF_SIZE
export _main
area text
_main:

;;...
```

```
M8c_EnableGInt                  ;Enable global interrupts
mov  A, >DMX_RAM_BUF            ;Save MSB of RAM buffer address
push A
mov  A, <DMX_RAM_BUF            ;Save LSB of RAM buffer address
push A
mov  A, DMX_RAM_BUF_SIZE        ;Save RW size parameter
push A
call DMX512Rx_SetRamBuffer      ;Set buffer
add  SP, -3                     ;Reset Stack
call DMX512Rx_Start             ;Turn on DMX512Rx user module
call DMX512Rx_EnableInt         ;Enable DMX512Rx interrupts
mov  X, 0
mov  A, 5
call DMX512Rx_SetStartSlotID    ;Redefine StartSlotID

.MainLoop:
call DMX512Rx_bGetSlotActivity ;Waiting for data...
cmp  A, 0
jz   .MainLoop
;-----------------------------
; Now data is ready
; User code goes here
;-----------------------------
jmp .MainLoop
```

The same code in C is as follows:

```
#include "PSoCAPI.h"    // PSoC API definitions for all user modules

#define DMX_RAM_BUF_SIZE 8
BYTE DMX_RAM_BUF[DMX_RAM_BUF_SIZE];
// ...

void main(void){
  M8C_EnableGInt;               // Enable global interrupts
  DMX512Rx_SetRamBuffer(DMX_RAM_BUF_SIZE, &DMX_RAM_BUF[0]);  //Set buffer
  DMX512Rx_Start();             // Turn on DMX512Rx User Module
  DMX512Rx_EnableInt();         // Enable DMX512Rx interrupts
  DMX512Rx_SetStartSlotID(5);   // Redefine StartSlotID
  while(1){
    while(!DMX512Rx_bGetSlotActivity());  // Wait for slot received
    //----------------------------
    // Now data is ready
    // User code goes here
    //----------------------------
} }
```

Configuration Registers

The DMX512Rx uses two digital PSoC blocks named RX and PWD. Each block is personalized and parameterized through registers. The following tables shows the register values as constants and the parameters as named bit-fields with brief descriptions. Symbolic names for these registers are defined in the user module instance C and assembly language interface files (the *.h* and *.inc* files).

Table 2. Block RX, Function Register, Bank 1

Bit	7	6	5	4	3	2	1	0
Value	InvertInput	BCEN	0	0	0	1	0	1

InvertInput allows to invert input data stream. This parameter is set in the Device Editor. BCEN gates the output onto the row broadcast bus line. This bit field is set in the Device Editor by directly configuring the broadcast line.

Table 3. Block RX, Input Register , Bank 1

Bit	7	6	5	4	3	2	1	0
Value	Input				RXClock			

The user module Input parameter in the Device Editor determines the value of the RXInput. RXClock selects the clock to drive the receiver timing.

Table 4. Block RX, Output Register, Bank 1

Bit	7	6	5	4	3	2	1	0
Value	ClockSync		0	0	0	0	0	0

The user module ClockSync parameter in the Device Editor determines the value of the ClockSync.

Table 5. Block RX, Shift Register (DR0), Bank 0

Bit	7	6	5	4	3	2	1	0
Value	RX Shift Register							

RX Shift Register: When a start bit is detected on the input, the RX state machine hardware generates a divide-by-8 bit clock that shifts data into this register.

Table 6. Block RX, Buffer Register (DR2), Bank 0

Bit	7	6	5	4	3	2	1	0
Value	RX Buffer Register							

RX Buffer Register: Data is transferred from the RX Shift register after the stop bit has been sampled.

Table 7. Block RX, Control Register (CR0), Bank 0

Bit	7	6	5	4	3	2	1	0
Value	0	Overrun	Framing	0	0	0	0	Start/Stop

Overrun is a flag that indicates that the RX Buffer register data is overwritten. Framing is a flag that indicates the stop bit was properly received. Start/Stop indicates that the DMX512Rx is enabled when set. It is modified using the DMX512RX API.

Table 8. Block PWD, Function Register, Bank 1

Bit	7	6	5	4	3	2	1	0
Value	InvertInput	BCEN	1	0	0	0	1	0

InvertInput allows to invert input data stream. This parameter is set in the Device Editor. BCEN gates the output onto the row broadcast bus line. This bit field is set in the Device Editor by directly configuring the broadcast line.

Table 9. Block PWD, Input Register , Bank 1

Bit	7	6	5	4	3	2	1	0
Value	Input				PWDClock			

The user module Input parameter in the Device Editor determines the value of the Input. PWDClock selects the clock to drive the receiver timing.

Table 10. Block PWD, Output Register, Bank 1

Bit	7	6	5	4	3	2	1	0
Value	ClockSync		0	0	0	0	0	0

The user module ClockSync parameter in the Device Editor determines the value of the ClockSync.

Table 11. Block PWD, Shift Register (DR0), Bank 0

Bit	7	6	5	4	3	2	1	0
Value	PWD Shift Register							

PWD Shift Register is the PWD Linear Feedback Shift register.

Table 12. Block PWD, Polynomial Register (DR1), Bank 0

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

Table 13. Block PWD, Compare Register (DR2), Bank 0

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

Table 14. Block PWD, Control Register (CR0), Bank 0

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	0	0	0	Start/Stop

Start/Stop indicates that the DMX512Rx is enabled when set. It is modified using the DMX512RX API.

Version History

Version	Originator	Description
1.0	DHA	Initial version

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.