SMIA 1.0 Part 2: CCP2 Specification

DISCLAIMER

History

Version	Date	Author	Status	Notes
1.0	30-June-04	Nokia and ST	Approved	

Table of contents

List of tables

List of figures

Acronyms Abbreviations and Definitions:

CCP2	Compact Camera Port 2
CCI	Camera Control Interface
EMC	Electro Magnetic Compatibility
EMI	Electro Magnetic Interference
EOF	End of Frame
EOL	End of Line
EXIF	Exchangeable Image File Format
Fps	Frames per second
FSP	False Synchronization Code Protection algorithm
I2C	Inter IC bus
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Expert Group
LSB	Least Significant Byte
LVDS	Low Voltage Differential Signalling
Mbps	Megabits per second
MSB	Most Significant Byte
PWB	Printed Wired Board
PLL	Phase Locked Loop
RAM	Random Access Memory
RGB	Color presentation (Red, Green Blue)
SOF	Start of Frame
SOL	Start Of Line
SubLVDS	Sub-Low Voltage Differential Signalling
SVGA	Super Video Graphics Array (800x600)
VGA	Video Graphics Array (640x480)
YUV	Color presentation (Y for chrominance, U&V for luminance)

Table 1: Acronyms

PREFACE

Specification Supersedes Earlier Documents

This document contains the draft SMIA Functional specification.

Following publication of the SMIA Standard, there may be future approved errata and/or approved changes to the standard prior to the issuance of another formal revision.

Incorporation of Engineering Change Requests (ECRs)

The following ECRs have been incorporated into this version of the specification:

ECR	DESCRIPTION

Table 2: ECR

1. Overview

The Compact Camera Port 2 (CCP2) specification defines an interface between digital camera module and mobile phone engine. The purpose of this document is to specify a standard interface between camera and phone engine. The CCP2 specification is a development version of current camera interface (CCP).

The mobile phone engine in this document means the hardware and software that performs essential core functions for telecommunication or application tasks. The engine of a mobile terminal includes hardware and the functions, which enable the basic operation of the mobile terminal. These include, for example, the printed circuit boards, RF components, basic electronics, and basic software, such as the digital signal processing software. Sensor module is required to automatically pad lines with dummy pixels to ensure that the number of pixels between the line start sync code for line m and the line end sync code for line m is a multiple

2. Overview of CCP2

The CCP2 specification defines standard data transmission and control interfaces between transmitter and receiver. Data transmission interface (referred as CCP2) is unidirectional differential serial interface with data and clock/strobe signals. Figure 1 illustrates connections between CCP2 transmitter and receiver, which typically are a camera module and a receiver module, part of the mobile phone engine.
The control interface (referred as CCI) is a bi-directional control interface compatible with I2C standard.

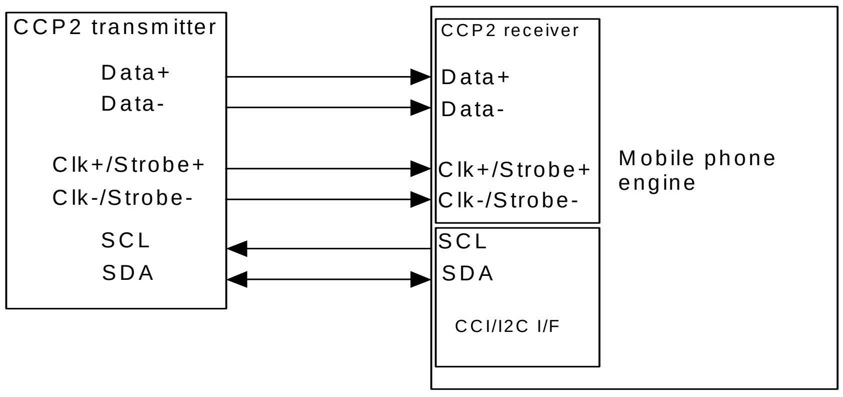


Figure 1. CCP2 and CCI interface between transmitter and receiver

CCP2 defines two options for data transfer between transmitter and receiver. The first one is normal data and clock. The second one is based on signaling scheme called data-strobe, which is a method for data transfer not needing a continuous clock signal. The clock is reconstructed at the receiving end from the data and strobe signals. The physical layer of CCP2 is based on signaling scheme called SubLVDS, which is current mode differential low voltage signaling method modified from the IEEE 1596.3 LVDS standard for reduced power consumption. Electrical specifications for the SubLVDS I/O's can be found from chapter 9. The use of data-strobe coding together with SubLVDS enables the use of high data rates with low EMI.

It is recommended that the clock for generating data and strobe signals is generated from host system clock using PLL or clock doubler. This simplifies greatly EMC design. Additional oscillators should be avoided in the camera module.

3. Signaling layer

3.1 Introduction

This specification defines two alternatives for signaling method to be used with CCP2, data-clock and data-strobe. The data-strobe signaling method is used e.g. in IEEE STD 1355-1995.

3.1.1 Data/clock signaling

CCP_DATA is a differential data signal, output from camera module. Data is written on each falling edge of CCP_CLK. Data format is in most of cases **bytewise (i.e. on 8-bit boundary) least significant bit (LSB) first.** The exceptions to this rule are presented in the data format section. When nothing is being transferred, CCP_DATA remains high, except in power shutdown (see chapter 10). Figure 2 illustrates the bytewise LSB first transmission

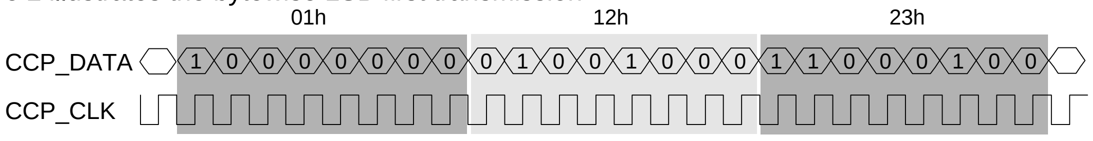


Figure 2. Transferring data sequence 011223h on CCP2.

CCP_CLK is a differential clock signal, output from camera module. Data is written in CCP_DATA on falling edge of the CCP_CLK. The receiving end should read the data on rising edge of the CCP_CLK. The clock signal may be free running or gated. For most cases free running clock is preferred due to simpler implementation in the transmitting end. However, in some cases gated clock may be better solution (see [chapter 7.11]). If gated transmission clock is used, clk remains high when nothing is being transferred, except in power shutdown (see chapter 10).

3.1.2 Data/Strobe signalling

The data-strobe coding consists of two parallel signals, data and strobe. The data signal carries the bit-serial data while the strobe signal state toggles whenever data signal does not change state. Thus, either the data signal or the strobe signal changes between two data bits. Both signals may not change simultaneously; it will be interpreted as an error. The signaling method is presented in the figure below.
The benefit of using data-strobe signaling is that there is no need for transferring continuous clock over the CCP2 bus. Also the frequency of the bus is divided by two. This makes the I/O cell development easier. In addition, EMI is reduced compared to normal data/clock signaling.

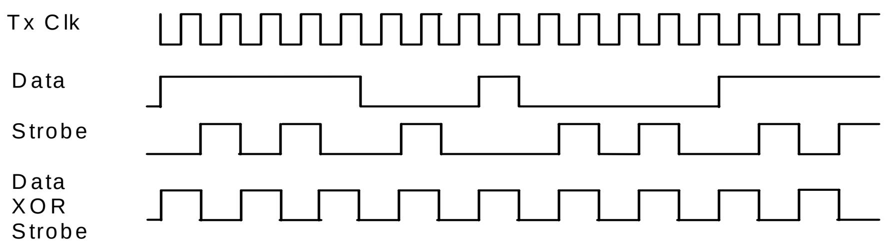


Figure 3. Data-strobe signaling

The data format for data-strobe is similar to data-clock signaling, data is sent bytewise LSB first.

The state of the data and strobe signals at the beginning of transmission are fixed. The state of data is logic high and the state of strobe is logic low.
The number of clock cycles between synchronization codes has to be even, both between SOL(or SOF)-EOL and EOL(EOF)-SOL.(SOF) This ensures synchronization is possible with minimum complexity to achieve the fastest possible implementation. Otherwise the synchronization detection would have to detect the phase of the data and adjust the data reception to the incoming data stream. This would require additional logic in the receiver.
The strobe signal can be gated when using the data/strobe signaling, but only if the number of clock cycles is even between synchronization codes.

3.1.2.1 Data/strobe phase relationship

At the beginning of transmission, the data should be initialized to ‘1’ and strobe to ‘0’. The number of transmission clock cycles between synchronization codes should be even. This ensures that for each synchronization code sequence FF0000h there will be corresponding strobe sequence of 55AAAAh as illustrated in the Figure 4 below.

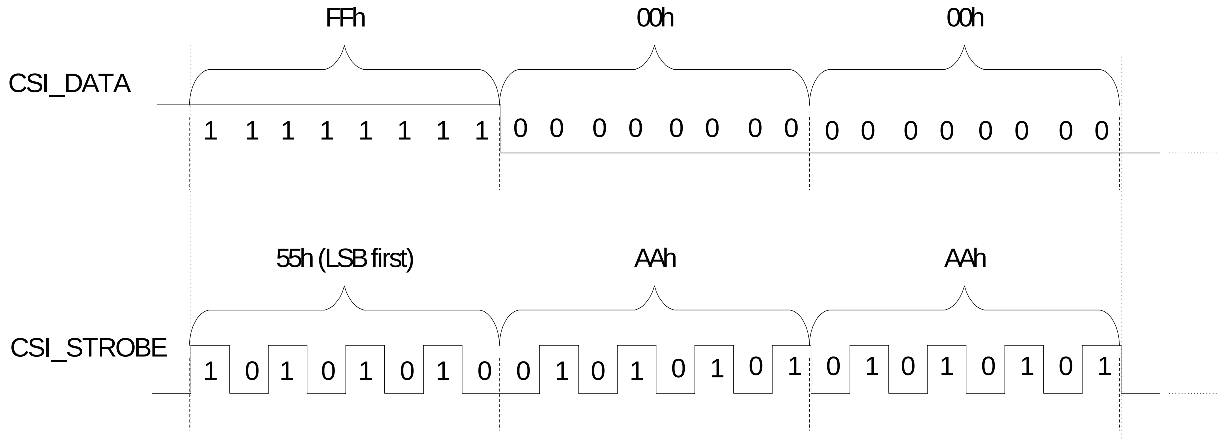


Figure 4. Data/strobe phase relationship

3.2 Definition of skew

The skew can be either the channel-to-channel skew, i.e. skew between data channel and clock/strobe channel, or complementary skew within signals in differential pair. The complementary skew is measured from differential zero to differential zero as illustrated in the Figure 5. The complementary skew is expressed as

$$t_{cmpskew} = |t_{HLA} - t_{LHB}| or |t_{HLB} - t_{LHA}|$$

where $t_{HLA/B}$ and $t_{LHB/A}$ are the propagation delays on outputs A and B for high to low and low to high.

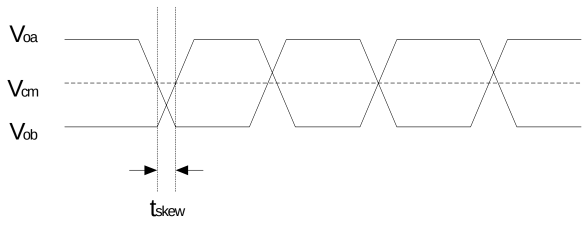


Figure 5. Complementary skew

The channel-to-channel skew is also measured from differential zero to differential zero as illustrated in the Figure 6. The channel-to-channel skew is expressed as

$$t_{chcskew} = \left| t_{diff}[m] - t_{diff}[n] \right|$$

where *m* is either one of the channels and *n* the other channel.

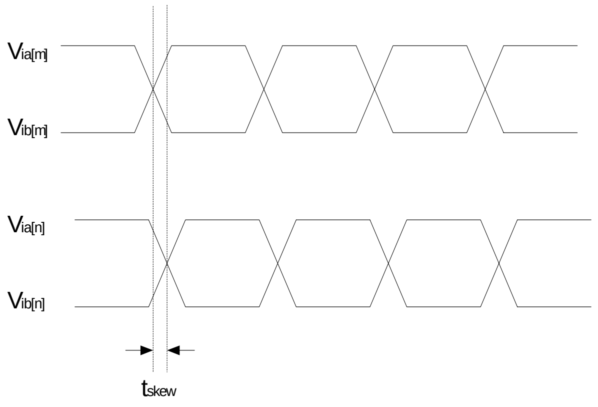


Figure 6. Channel-to-channel skew

The nominal values for these two skews are specified in the Table 3.

Symbol	Parameter	Min	Max	Unit
$t_{cmpskew}$	$\left\|t_{HLA} - t_{LHB}\right\| or \left\|t_{HLB} - t_{LHA}\right\|$, differential skew		50	ps
$t_{chcskew}$	$\left\|t_{diff}[m] - t_{diff}[n]\right\|$, channel-to-channel skew		100	ps

Table 3: Skew definitions

3.3 CCP2 classification

The transmitter/receiver pairs are classified in three classes, classified by the data transfer capacity and signaling method. This is illustrated in the Table 4.

Class	Data transfer capacity (sustain data rate)	Signaling method
Class 0	<208 Mbit/s	Data/Clock
Class 1	208 Mbit/s to 416 Mbit/s	Data/Strobe
Class 2	416 Mbit/s to 650 Mbit/s	Data/Strobe

Table 4: CCP2 classification

For each class there is a different set of parameters for jitter and skew to be fulfilled for transmitter and receiver. The link timing budget for each class is presented in the Table 5.

	Data Clock		**Data Strobe**			
	Class 0 208		**Class 1 416**		**Class 2 650**	
	Jitter [ps]	**Skew [ps]**	**Jitter [ps]**	**Skew [ps]**	**Jitter [ps]**	**Skew [ps]**
TX Jitter	5% 240		5% 120		5% 77	
TX clock duty cycle variance (+/-5%)		480				
TX encoder & driver	200	300	200	250	200	125
TX package		100		100		100
Socket, PCB, Flex, Flex Connector		400		350		250
RX Setup time	1 ns					
TX hold time	2.4 ns					
RX Data & Strobe BOTH Stable				1350		780
Sum	**200**	**1280**	**200**	**2050**	**200**	**1255**
Total (NOT inc TX Jitter)	**1480**		**2250**		**1455**	
Peak Bit Rate	**218.4**		**444.4**		**687.3**	
Overall Total (TX Jitter)	**1720**		**2370**		**1532**	
Max Usable Bit Rate	**208.0**		**421.9**		**652.8**	
Target Period	**4808**		**2404**		**1538**	

Table 5: CCP2 link timing budget

4. Camera Control Interface (CCI)

The CCI is an I2C Fast-mode compatible interface for controlling the transmitter. The CCP2 receiver is always a master and CCP2 transmitter always a slave in the CCI bus. CCI is capable of handling several slaves in the bus, but multi-master mode is not supported. Typically no other devices than CCP2 receiver and transmitter are connected to the CCI bus. This makes a pure SW implementation possible.

Typically the CCI is separate from the system I2C bus, but I2C compatibility ensures that it is also possible to connect the transmitter to system I2C bus. CCI is a subset of I2C protocol including the minimum combination of obligatory features for I2C slave device specified in the I2C specification. Therefore transmitters complying with the CCI specification can also be connected to system I2C bus. However, it has to be taken care that the I2C masters do not try to utilize those I2C features, which are not supported in transmitters complying with the CCI specification. Each transmitter conforming the CCI specification may have additional features implemented to support I2C, but that is dependent of implementation. Further details can be found from particular device’s data sheet.

The CCI defines an additional layer of data protocol on top of I2C. The data protocol is presented in the following chapters.

The control interface is either CCI or genuine I2C; all other means for control data transfer are not CCP2/CCI compliant.

4.1 Data transfer protocol

The data transfer protocol is according to I2C standard. The START, REPEATED START and STOP conditions as well as data transfer protocol are specified in the I2C specification [4].

4.1.1 Message type

As I2C standard does not define the message formats, the CCI defines an additional layer above I2C data transfer protocol. A basic CCI message consists of START condition, slave address with read/write bit, acknowledge from slave, sub address (index) for pointing register inside the slave device, ack signal from slave, in write operation data byte from master, ack/negative ack from slave and STOP condition. In read operation data byte comes from slave and ack/negative ack from master. This is illustrated in the Figure 7 below.
The slave address in the CCI can be either 7-bit or 10-bit. CCI itself does not explicitly define the addressing scheme; it is dependent on the slave device in question.
The CCI supports 8-bit index with 8-bit data or 16-bit index with 8-bit data. The slave device in question defines what is the message type to be used.

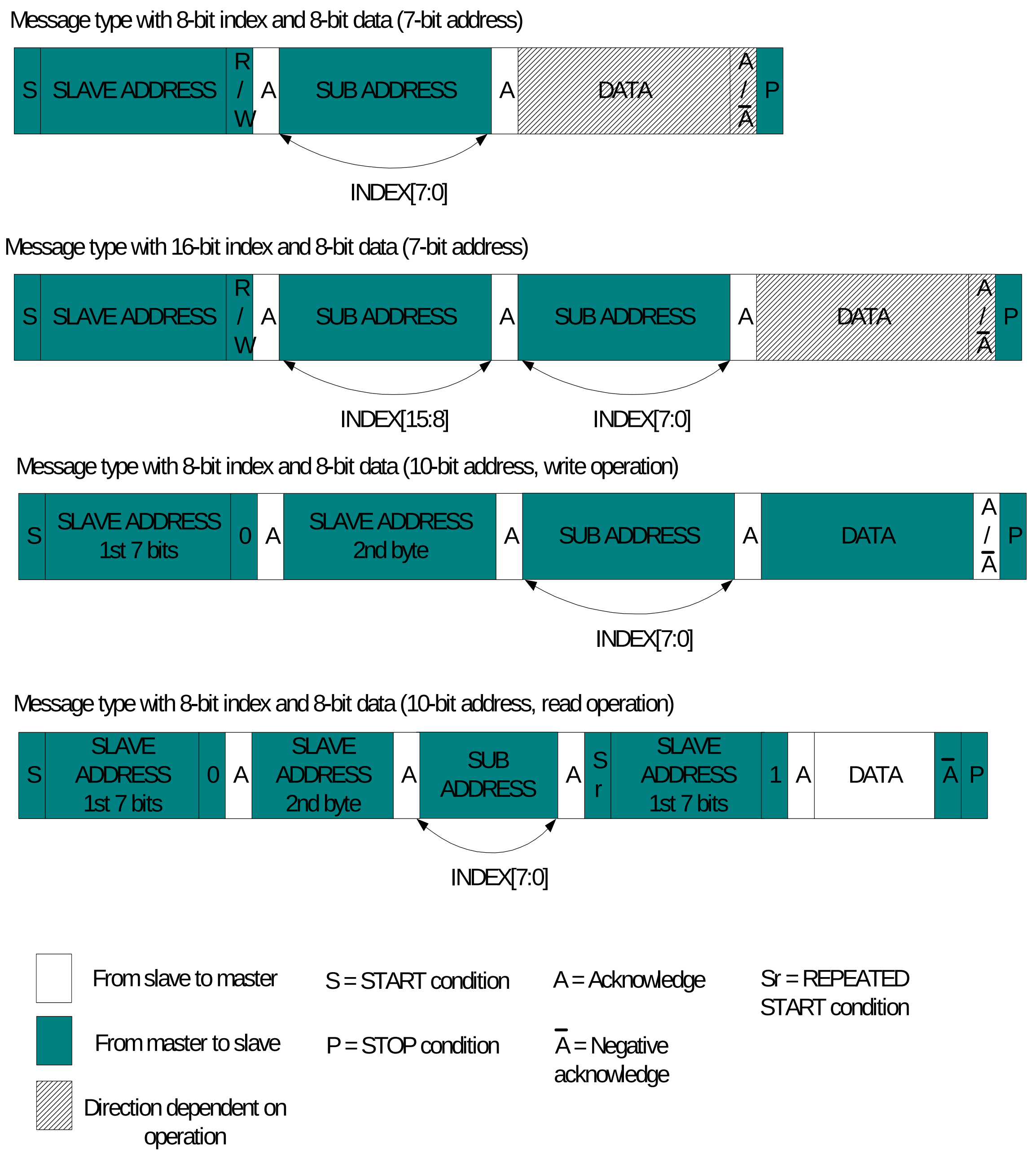


Figure 7. CCI message

4.1.2 Read/write operations

The CCI compatible device must be able to support four different read operations and two different write operations; single read from random location, sequential read from random location, single read from current location, sequential read from current location, single write to random location and sequential write starting from random location. The read/write operations are presented in the following chapters.
The index in the slave device has to be auto incremented after each read/write operation. This is also explained in the following chapters.

4.1.2.1 Single read from random location

In single read from random location the master does a dummy write operation to desired index, issues a repeated start condition and then addresses the slave again with read operation. After acknowledging its slave address, the slave starts to output data onto SDA line. This is illustrated in the Figure 8 below. The master terminates the read operation by setting a negative acknowledge and stop condition.

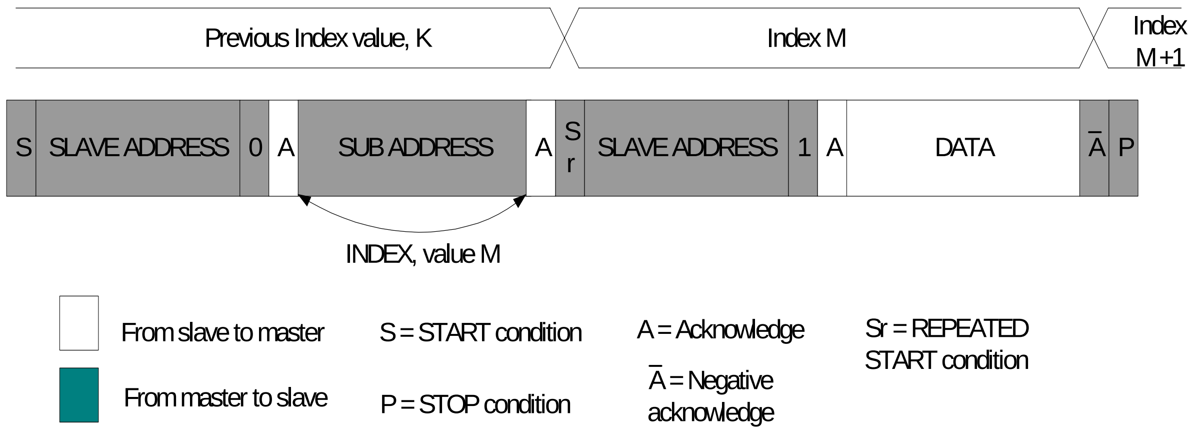


Figure 8. CCI single read from random location

4.1.2.2 Single read from current location

It is also possible to read from last used index by addressing the slave with read operation. The slave responses by setting the data from last used index to SDA line. This is illustrated in the Figure 9 below. The master terminates the read operation by setting a negative acknowledge and stop condition.

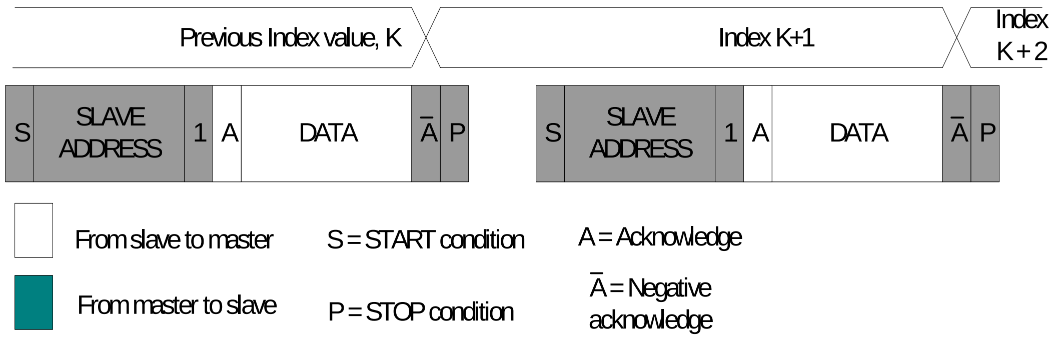


Figure 9. CCI single read from current location

4.1.2.3 Sequential read starting from random location

The sequential read starting from random location is illustrated in the Figure 10 below. The master does a dummy write to the desired index, issues a repeated start condition after acknowledge from slave and addresses the slave again with read operation. If a master issues acknowledge after received data, it acts as a signal to slave that the read operation continues from next index. When master has read the last data byte, it issues a negative acknowledge and stop condition.

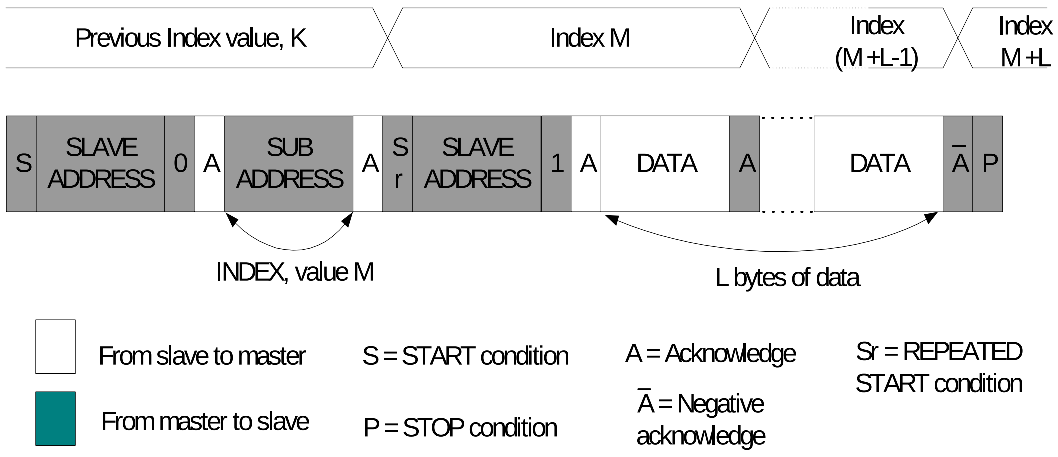


Figure 10. CCI sequential read starting from random location

4.1.2.4 Sequential read starting from current location

The sequential read starting from random location is similar to sequential read from random location. Only exception is that there is no dummy write operation. This is illustrated in the Figure 11 below. The master terminates the read operation by setting a negative acknowledge and stop condition.

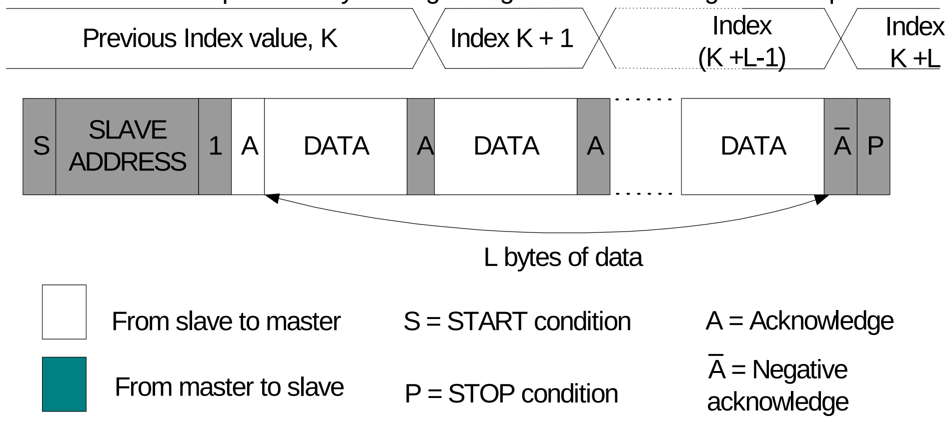


Figure 11. CCI sequential read starting from current location

4.1.2.5 Single write to random location

The write operation to random location is illustrated in the Figure 12. The master issues a write operation to the slave, sets the index and data correspondingly after the slave has acknowledged. The write operation is terminated with stop condition from the master.

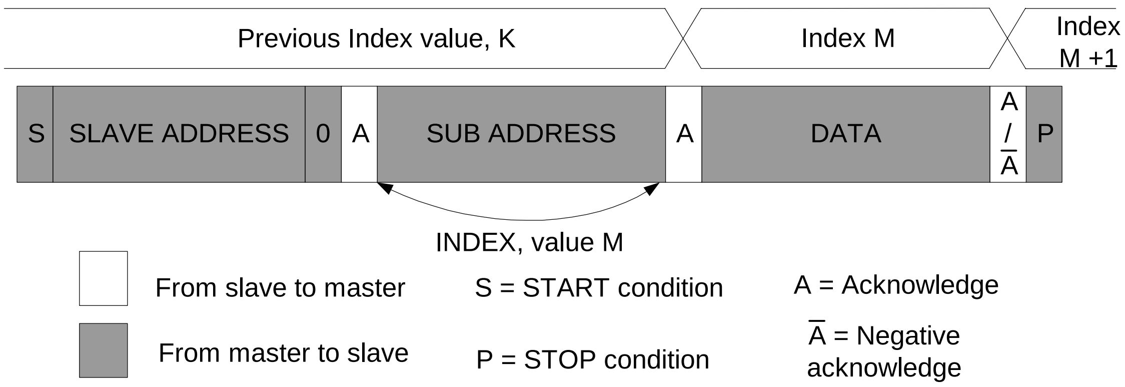


Figure 12. CCI single write to random location

4.1.2.6 Sequential write

The sequential write is illustrated in the Figure 13 below. The slave auto increments the index after each data byte. The sequential write operation is terminated with stop condition from the master.

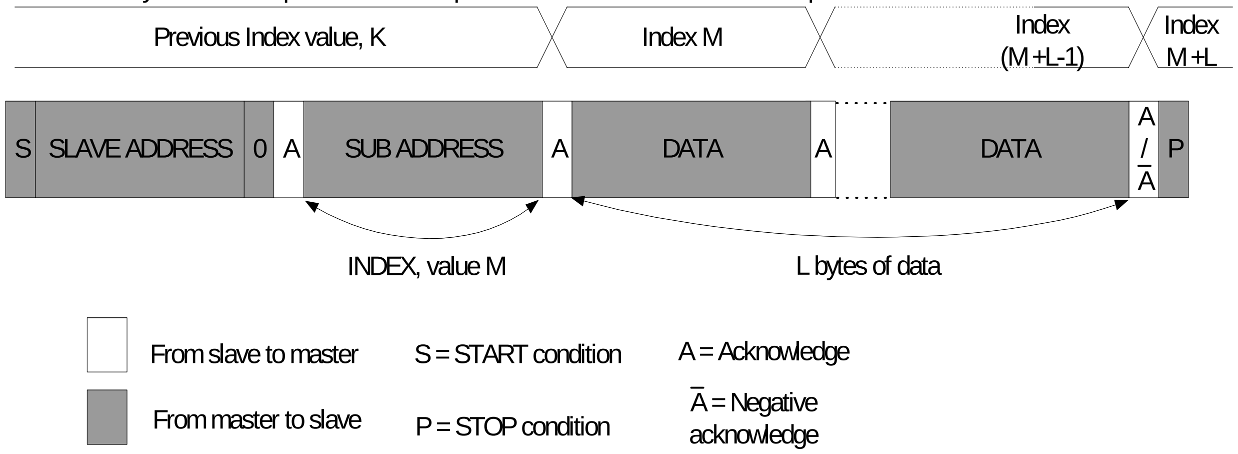


Figure 13. CCI sequential write starting from random location

4.2 Electrical specifications and timing for I/O stages

The electrical specification and timing for I/O stages conform I2C standard for Standard- and Fast-mode devices. Information presented in the tables below is a direct copy of I2C standard.

Parameter	Symbol	Standard-mode		Fast-mode		Unit
		Min.	Max.	Min.	Max.	
LOW level input voltage	V_{IL}	-0.5	0.3 V_{DD}	-0.5	0.3 V_{DD}	V
HIGH level input voltage	V_{IH}	0.7 V_{DD}	Note 1	0.7 V_{DD}	Note 1	V
Hysteresis of Schmitt trigger inputs V_{DD} > 2V V_{DD} < 2V	V_{hys}	 N/A N/A	 N/A N/A	 0.05 V_{DD} 0.1 V_{DD}	 - -	 V V
LOW level output voltage (open drain) at 3mA sink current V_{DD} > 2V V_{DD} < 2V	 V_{OL1} V_{OL3}	 0 N/A	 0.4 N/A	 0 0	 0.4 0.2 V_{DD}	 V V
HIGH level output voltage	V_{OH}	N/A	N/A	0.8 V_{DD}		V
Output fall time from V_{IHmin} to V_{ILmax} with bus capacitance from 10 pF to 400 pF	t_{of}	-	250	20+0.1 C_B Note2	250	ns
Pulse width of spikes which must be suppressed by the input filter	t_{sp}	N/A	N/A	0	50	ns
Input current each I/O pin with an input voltage between 0.1 V_{DD} and 0.9 V_{DD}	I_I	-10	10	-10 Note 3	10 Note 3	μA
Input/Output capacitance (SDA)	$C_{I/O}$		8		8	pF
Input capacitance (SCL)	C_I		6		6	pF

Table 6: CCI I/O characteristics

Note 1	*Maximum $V_{IH} = V_{DDmax} + 0.5V$*
Note 2	*C_B = capacitance of one bus line in pF*
Note 3	*I/O pins of Fast-mode devices must not obstruct the SDA and SCL line if V_{DD} is switched off*

Parameter	Symbol	Standard-mode		Fast-mode		Unit
		Min.	Max.	Min.	Max.	
SCL clock frequency	f_{SCL}	0	100	0	400	kHz
Hold time (repeated) START condition. After this period, the first clock pulse is generated	$t_{HD:STA}$	0.4	-	0.6	-	μs
LOW period of the SCL clock	t_{LOW}	4.7	-	1.3	-	μs
HIGH period of the SCL clock	t_{HIGH}	4.0	-	0.6	-	μs
Setup time for a repeated START condition	$t_{SU;STA}$	4.7	-	0.6	-	μs

Data hold time	$t_{HD;DAT}$	0 Note 2	3.45 Note 3	0 Note2	0.9 Note 3	µs
Data set-up time	$t_{SU;DAT}$	250	-	100 Note 4	-	ns
Rise time of both SDA and SCL signals	t_r	-	1000	20+0.1 C_B Note 5	300	ns
Fall time of both SDA and SCL signals	t_f	-	300	20+0.1 C_B Note 5	300	ns
Set-up time for STOP condition	$t_{SU;STO}$	4.0	-	0.6	-	µs
Bus free time between a STOP and START condition	t_{BUF}	4.7	-	1.3	-	µs
Capacitive load for each bus line	C_B	-	400	-	400	pF
Noise margin at the LOW level for each connected device (including hysteresis)	V_{nL}	0.1 V_{DD}	-	0.1 V_{DD}	-	
Noise margin at the HIGH level for each connected device (including hysteresis)	V_{nH}	0.2 V_{DD}	-	0.2 V_{DD}	-	pF

Table 7: CCI timing specification

Note 1	*All values referred to V_{IHmin} = 0.9 V_{DD} and V_{ILmax} = 0.1 V_{DD}*
Note 2	*A device must internally provide a hold time of at least 300 ns for the SDA signal (referred to the V_{IHmin} of the SCL signal) to bridge theundefined region of the falling edge of SCL*
Note 3	*The maximum $t_{HD;DAT}$ has only to be met if the device does not the LOW period (t_{low})of the SCL signal*
Note 4	*A Fast-mode I2C-bus device can be used in a Standard-mode I2C-bus system, but the requirement $t_{SU;DAT} \geq 250$ ns must be then met. This will be automatically the case if the device does not stretch the LOW period of the SCL signal. If such device does stretch the low period of SCL signal, it must output the next data bit to the SDA line t_{rMAX} + $t_{SU;DAT}$ = 1000 + 250 = 1250 ns (according to the Standard-mode I2 bus specification) before the SCL line is released.*
Note 5	*C_B = total capacitance of one bus lin in pF.*

The timing is illustrated in the Figure 14 below.

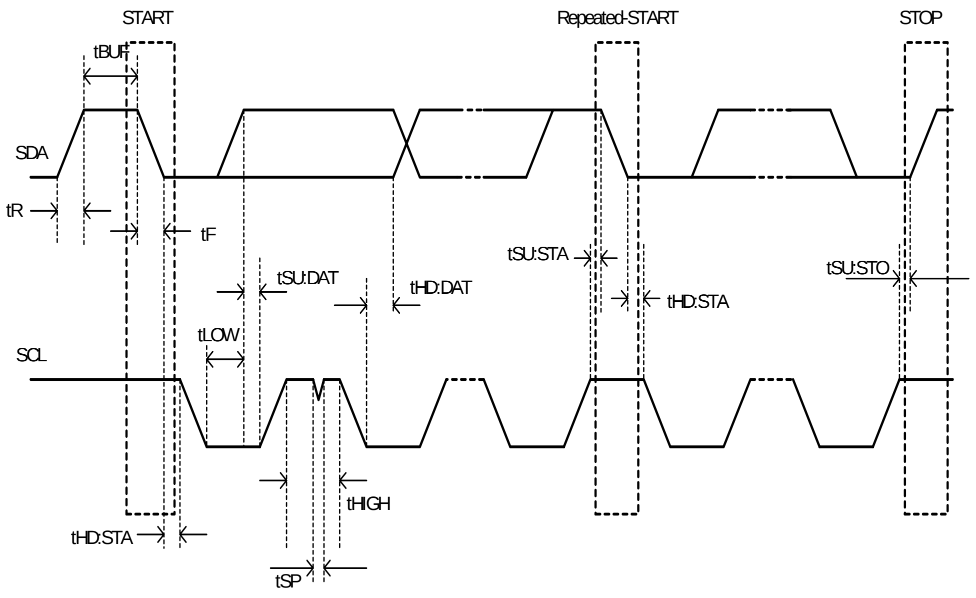


Figure 14. CCI timing

4.3 Unsupported I2C features in CCI

The differences between I2C and CCI are presented below.

General call address & other "special" addresses, not supported
General call address (0000 0000), start byte (0000 0001) transmitted as an address, CBUS and other different bus format addresses (0000 001X, 0000 010X) and the reserved addresses (0000 011X, 1111 1XXX) are not used in CCI and are therefore unsupported. See also *high speed mode*.

High speed mode, not supported
The CCI does not support the Hs (= high speed) mode, signified by the first address byte being 0000 1XXX). See also *general call address*.

Start byte, not supported
Start condition is implied as specified in I2C standard. Start byte is not supported.

5. Overview of image data formats and definitions

The applicable data formats specified for CCP2 are listed in Table 8. Abbreviations for each format are also defined.

Data type	**Abbreviation**
YUV 422 image data	YUV422
YUV 420 image data	YUV420
RGB 888 image data	RGB888
RGB 565 image data	RGB565
RGB 444 image data	RGB444
Raw bayer, 6-bit image data	RAW6
Raw bayer, 7-bit image data	RAW7
Raw bayer, 8-bit image data	RAW8
Raw bayer, 10-bit image data	RAW10
Raw bayer, 12-bit image data	RAW12
JPEG 8-bit data	JPEG8

Table 8: CCP2 data formats

Each transmitter may have a chosen subset of the specified formats implemented, i.e. implementing all formats is not mandatory for CCP2 compatible transceiver, but CCP2 compatible receiver has to implement all formats specified above.

5.1 Frame structure

The structure of an image frame is presented in the Figure 15 below. The period between line end code and new line start code is called line blanking period. Similarly, the time between frame end code and new frame start code is called frame blanking period. The period between line end code and new line start code is called line blanking period. Similarly, the time between frame end code and new frame start code is called frame blanking period. The line blanking period may vary, the receiver should be able to cope with line blanking period set to zero. Transmitter defines the minimum time for frame blanking period. It is recommended that frame blanking period duration is programmable in the transmitter.
The size of one image line shall be a multiple of 32 bits.

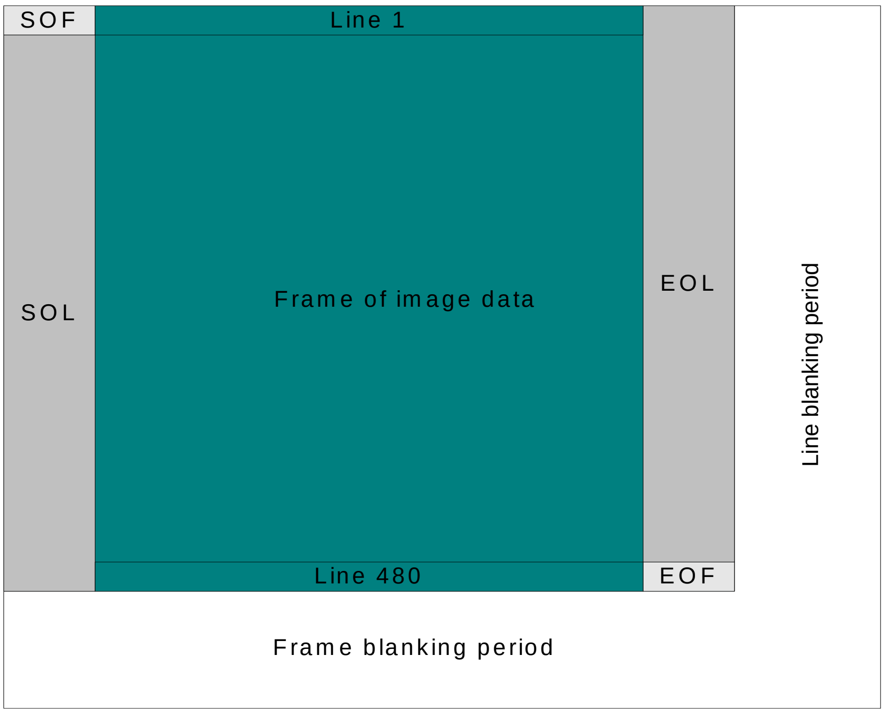


Figure 15. Frame structure in VGA case

5.2 Synchronization

Each image frame begins with frame start synchronization code and ends with frame end synchronization code. Each line inside the frame begins with line start synchronization code and ends with line end synchronization code. In the beginning of frame end in the end of frame line synchronization codes are replaced by the frame synchronization codes. Synchronization signal usage is shown in Figure 16.
Bit order of the synchronization codes is the same as for data, bytewise LSB first. This is illustrated in Figure 17.
If necessary, the clock signal may remain active in between line end/start and / or frame end/start (i.e. during blanking periods). This has to be clearly indicated in the transmitter specification.

Code	Value
Line start code	FF_H 00_H 00_H $X0_H$ (X = channel number 0 to 7)
Line end code	FF_H 00_H 00_H $X1_H$
Frame start code	FF_H 00_H 00_H $X2_H$
Frame end code	FF_H 00_H 00_H $X3_H$
Logical channels	FF_H 00_H 00_H $0X_H$ to FF_H 00_H 00_H $7X_H$

Table 9: Synchronization codes

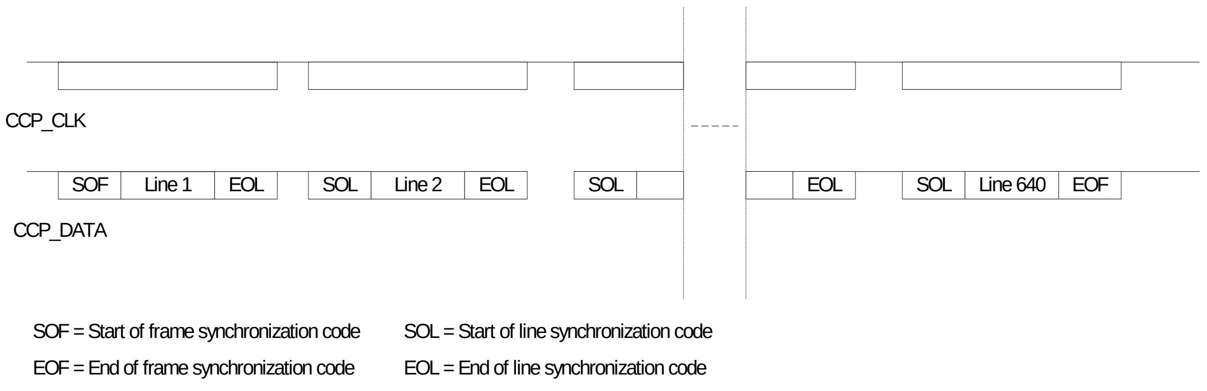

CCP_CLK
CCP_DATA
SOF
Line 1
EOL
SOL
Line 2
EOL
SOL
EOL
SOL
Line 640
EOF
SOF = Start of frame synchronization code
SOL = Start of line synchronization code
EOF = End of frame synchronization code
EOL = End of line synchronization code

Figure 16. Use of CCP2 synchronization codes (VGA system example)

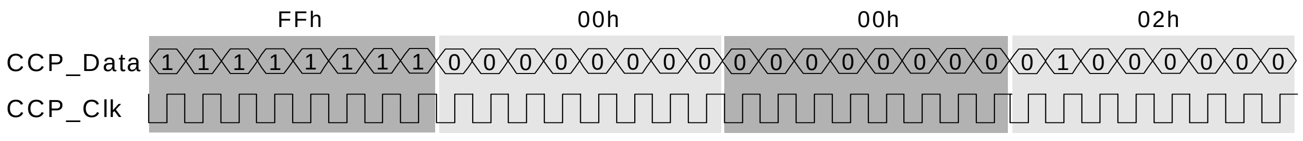


Figure 17. Bit order in synchronization codes (frame start code as an example

5.3 Logical (DMA) channels

The purpose of logical channels is to separate different data flows, which are interleaved in the data stream. The DMA channel identifier number is directly encoded in the 4-byte CCP embedded sync codes. The CCP receiver will monitor the DMA channel identifier and de-multiplex the interleaved video streams to their appropriate DMA channel. A maximum of 8 data streams is supported. Valid channel identifiers are 0 to 7. The DMA channel identifier must be fully programmable to allow the host to configure which DMA channels the different video data stream use. The principle of logical channels is presented in the Figure 18.

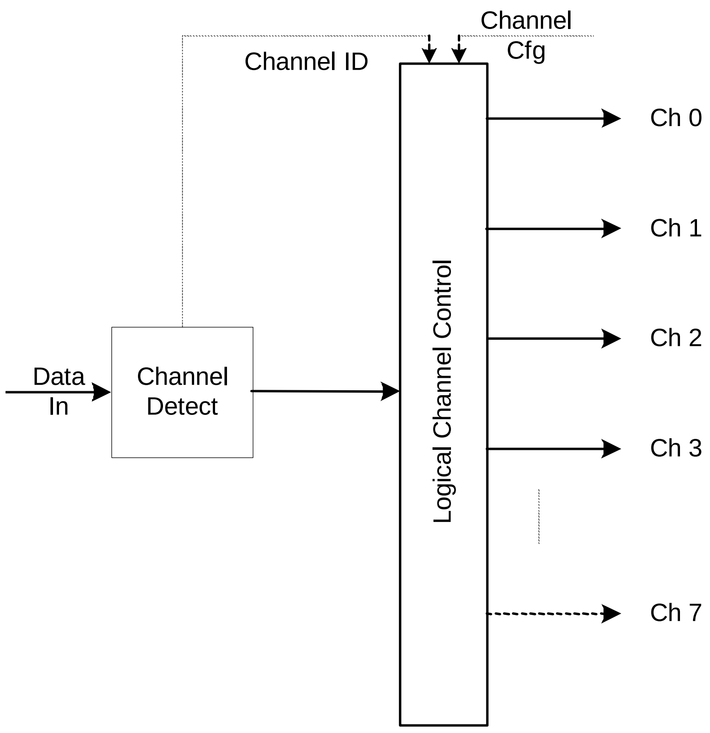


Figure 18. Logical Channel Block Diagram

The channel identifier is a part of CCP2 synchronization code, upper four bits of last byte of synchronization code. Figure 19 illustrates the synchronization code with logical channel identifiers.

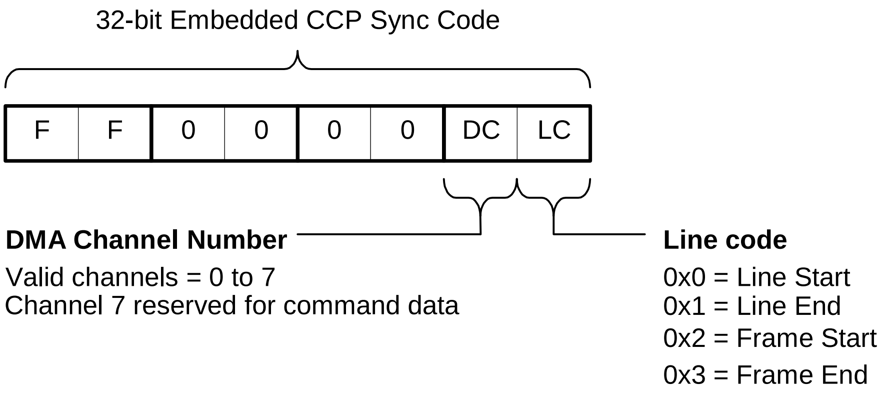


Figure 19 Modified CCP2 synchronization codes

The Figure 20 illustrates an example data stream utilizing logical channel support.

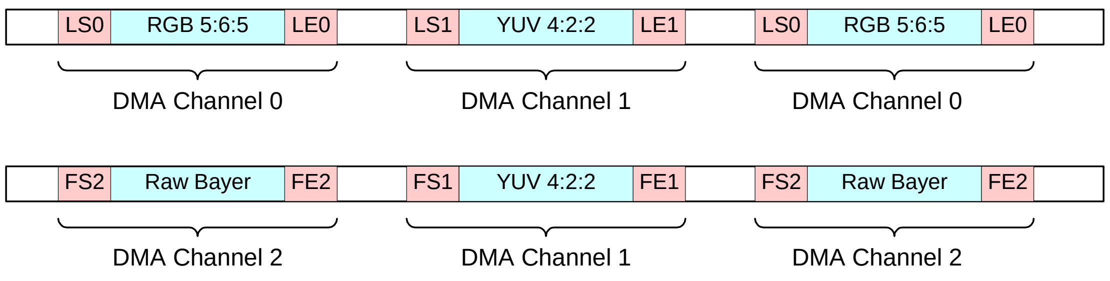


Figure 20 Interleaved CCP (Video) Data Streams Examples

5.4 False synchronization code protection

CCP2 receiver compares the incoming data bit by bit (i.e. bitwise, not bytewise). This may lead to situation where synchronization code appears inside image data and causes false synchronization to happen. In order to avoid false synchronization, the sequence FF 00 00h is forbidden inside the image data. Meaning, having 8 ones followed by 16 zeros is not allowed. For example sequences like 1F F0 00 0Ah and 01 FE 00 01h are forbidden. For each data format there is a recommendation how to avoid the false synchronization.

5.5 Embedded information

It is possible to embed extra lines containing additional information to the beginning and to the end of each picture frame as presented in the Figure 21. If there exists embedded information, it has to be clearly defined in the transmitter specification. In addition, the embedded information has to conform to general CCP2 requirements listed below.

- The length of the embedded information line has to be the same as the length of image data line. If the embedded information line is broader than image data line, the embedded information line has to be split in two different lines.
- The false synchronization code protection has to be implemented in the embedded information.
- Each line must have equal length

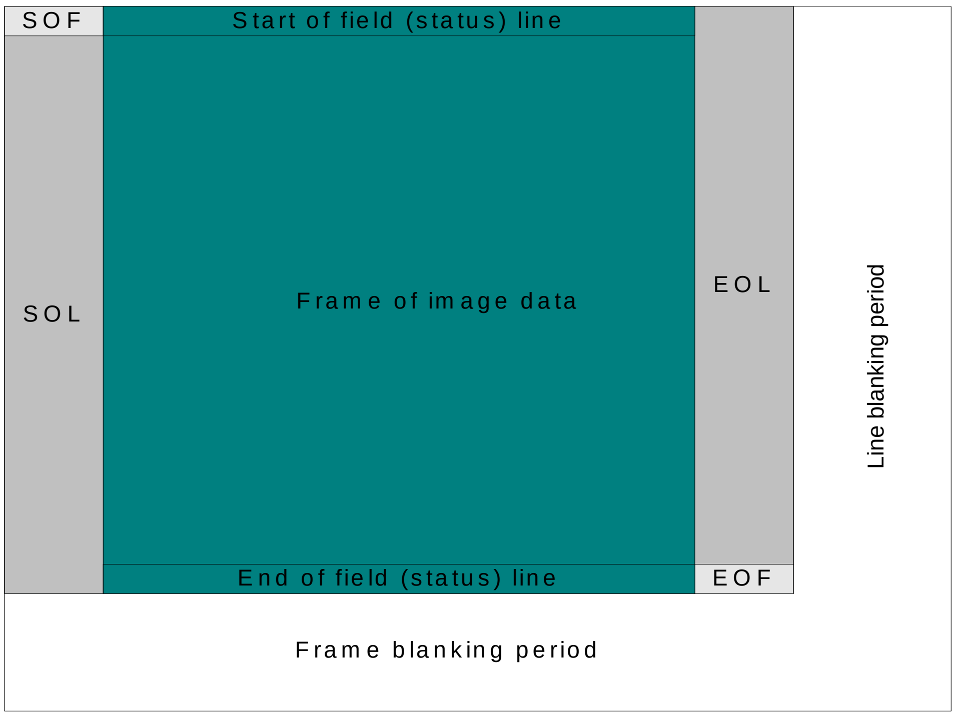


Figure 21. Frame structure with status lines at the beginning and in the end of frame

5.6 Checksum generation

To detect possible errors in transmission, a checksum is calculated over each data line, except JPEG8 for which the checksum is calculated over frame. The checksum is realized as 16-bit CRC. The generator polynomial is x^16+x^5+1. The checksum sequence is then padded with 16-bit padding sequence (101010…10) to fulfill the 32-bit word alignment requirement in the CCP2 link.
The transmission of checksum is illustrated Figure 22 in the below.

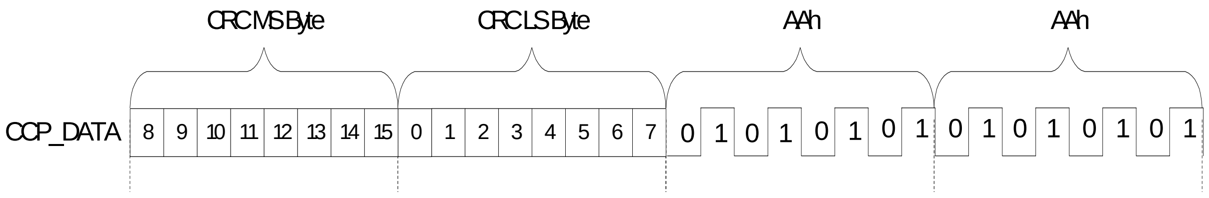


Figure 22. Checksum transmission

The 32-bit checksum sequence is transmitted after line/frame end synchronization code.

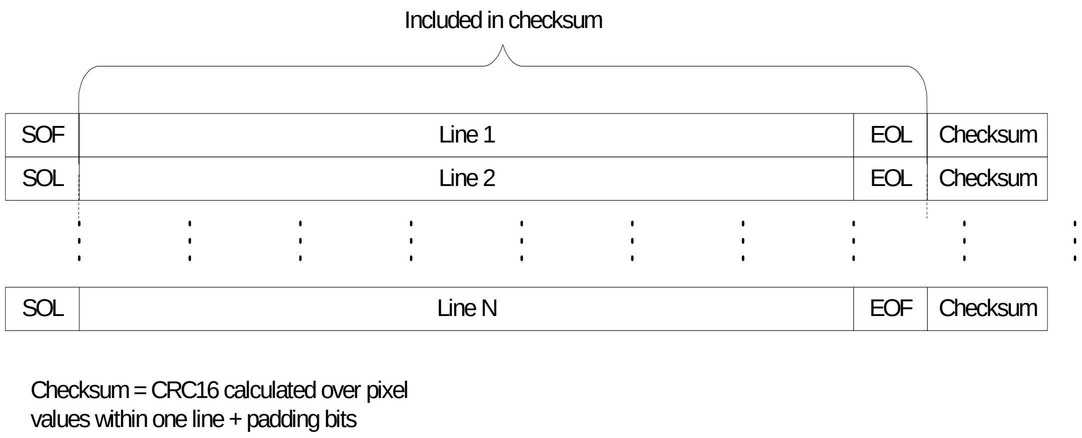


Figure 23. Checksum generation for line data

For JPEG8 the checksum is calculated over whole frame after FSP encoding. The FSP decoding in the receiver must happen then after checksum validation.

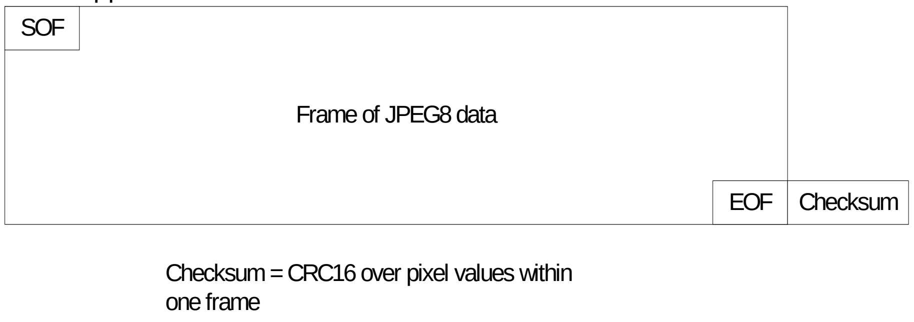


Figure 24. Checksum generation for JPEG8 data

An example CRC implementation is presented in the Figure 25 below. The CRC shift register is initialized to 0x00 at the beginning of line/frame. The image data comes from output shift register bitwise. Each bit is fed to the CRC shift register before it is passed forward to output. After all pixels in line/frame including the line/frame end synchronization code has passed the output shift register, the CRC shift register contains the checksum. The checksum is then padded and sent over CCP2 bus to the receiver to verify that no errors have occurred in the transmission.

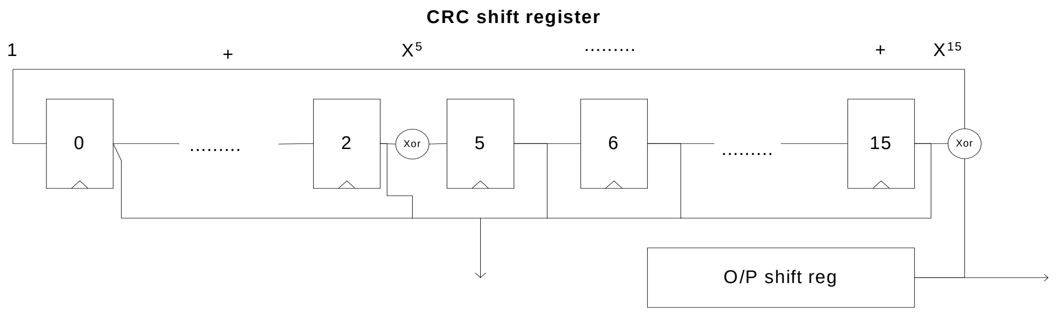


Figure 25. CRC calculation

6. Color spaces

6.1 Definition of RGB color space

In this specification, abbreviation RGB means the nonlinear sR'G'B' color space in 8-bit representation, based on definition of sRGB in IEC 61966 standard.
The 8-bit representation results as RGB888. The conversion to more commonly used RGB565 format is achieved by scaling the 8-bit values to 5 bits (blue and red) and 6 bits (green). The scaling can be done by simply dropping the LSBs.

6.2 Definition of YUV color space

In this specification, abbreviation YUV means the 8-bit gamma corrected $Y'C_BC_R$ colorspace defined in ITU-R BT601.4.

7. Data formats

For each data format there is presented how the data is organized on the bus. For clarity, the figures describe the Data/Clock signaling method. The Data/Strobe signaling method is essentially the same except there will be no clock signal but strobe signal. Since in the Data/Strobe signaling method the signal behavior is dependent on data, the Data/Clock signaling method is used in the examples.

7.1 YUV422

YUV422 data transmission is performed by transmitting UYVY… sequence. Each line is separated by line start / end synchronization codes. This sequence is illustrated in Figure 26.
Bit order in transmission follows the general CCP2 rule, LSB first. This is illustrated in Figure 27.
For YUV422 data the suggested way to solve the false synchronization code issue is to forbid value 0 from appearing in Y, U or V bytes. Minimum value for Y, U and V is thus 01h (U and V maximum range will then be –127 to 127, please see chapter 6.2 for reference).

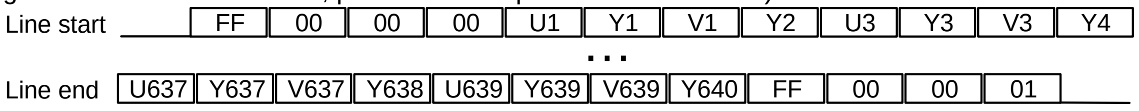


Figure 26. YUV422 transmitted bytes in VGA image case

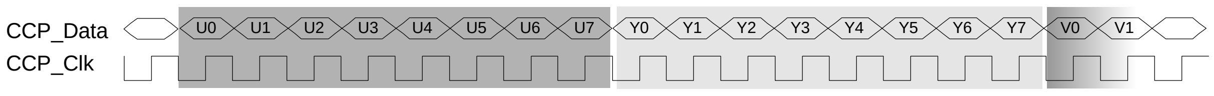


Figure 27. YUV422 transmission on CCP2 bus, bitwise illustration

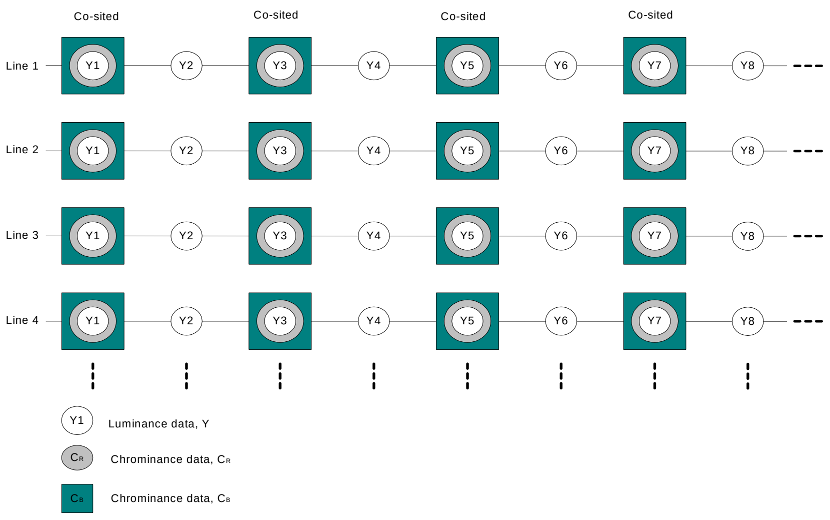


Figure 28. U and V pixel spatial alignment in YUV422 data

The pixel spatial alignment is the same as in CCIR-656 standard. The frame format for YUV422 is presented in the Figure 29.

SOF	U	Y	V	Y	U	Y	V	Y	SOF
	U	Y	V	Y	U	Y	V	Y	
	U	Y	V	Y	U	Y	V	Y	
	U	Y	V	Y	U	Y	V	Y	
	U	Y	V	Y	U	Y	V	Y	
SOL	U	Y	V	Y	U	Y	V	Y	SOL
	U	Y	V	Y	U	Y	V	Y	
	U	Y	V	Y	U	Y	V	Y	
	U	Y	V	Y	U	Y	V	Y	
	U	Y	V	Y	U	Y	V	Y	

Figure 29. Frame format in YUV422 data

7.2 YUV420

YUV420 data transmission is performed by transmitting UYY… / VYY… sequences in odd / even lines. U component is transferred in odd lines (1,3,5…) and V component is transferred in even lines (2,4,6…). Each line is separated by line start / end synchronization codes. This sequence is illustrated in Figure 30.

Bit order in transmission follows the general CCP2 rule, LSB first. This is similar to YUV422 transmission illustrated in Figure 27.

For YUV420 data the suggested way to solve the false synchronization code issue is to forbid value 0 from appearing in Y, U or V bytes. Minimum value for Y, U and V is thus 01h (U and V maximum range will then be –127 to 127, please see chapter 6.2 for reference).

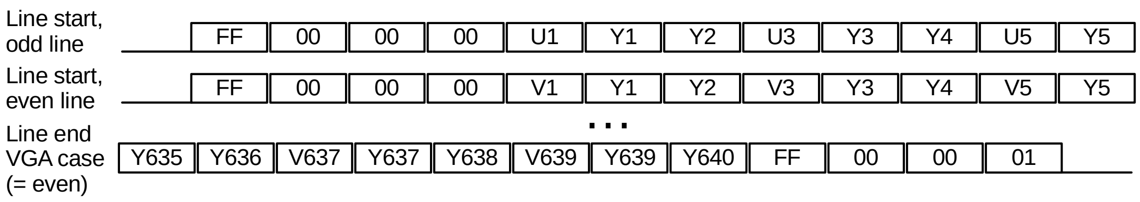

Line start, odd line
FF 00 00 00 U1 Y1 Y2 U3 Y3 Y4 U5 Y5
Line start, even line
FF 00 00 00 V1 Y1 Y2 V3 Y3 Y4 V5 Y5
. . .
Line end VGA case (= even)
Y635 Y636 V637 Y637 Y638 V639 Y639 Y640 FF 00 00 01

Figure 30. YUV420 transmitted bytes in VGA image case

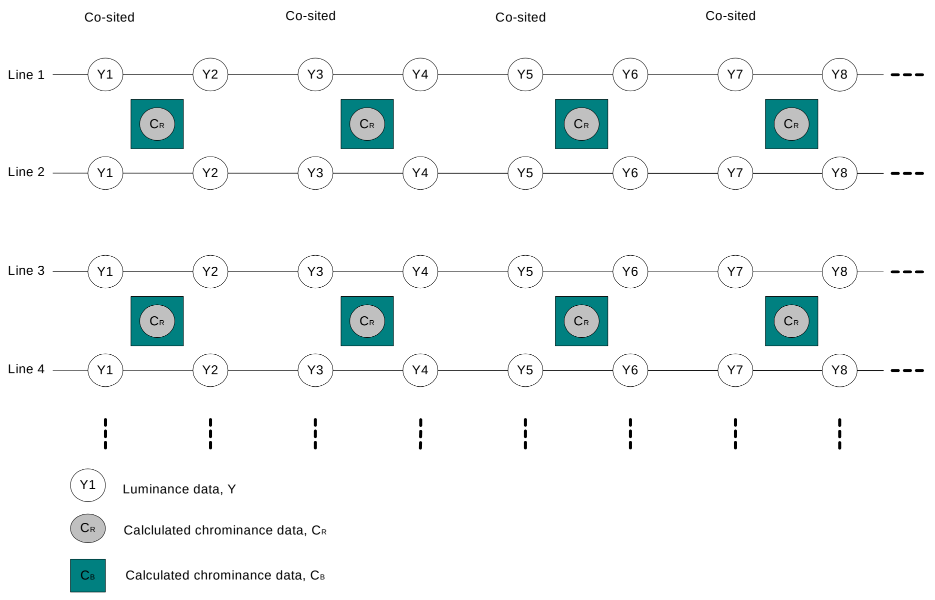
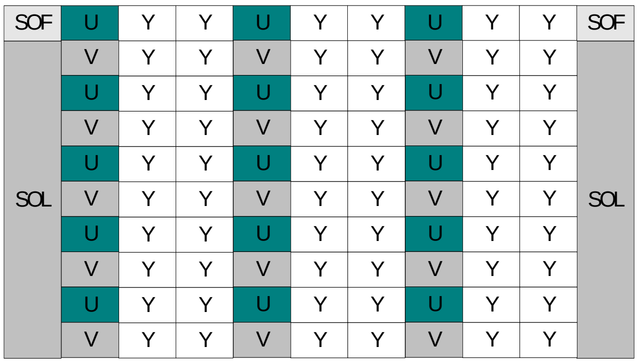


Figure 31. U **and V pixel spatial alignment in YUV420 data**

The pixel spatial alignment is the same as used in H.261, H.263 and MPEG1.



SOF	U	Y	Y	U	Y	Y	U	Y	Y	SOF
	V	Y	Y	V	Y	Y	V	Y	Y	
	U	Y	Y	U	Y	Y	U	Y	Y	
	V	Y	Y	V	Y	Y	V	Y	Y	
	U	Y	Y	U	Y	Y	U	Y	Y	
SOL	V	Y	Y	V	Y	Y	V	Y	Y	SOL
	U	Y	Y	U	Y	Y	U	Y	Y	
	V	Y	Y	V	Y	Y	V	Y	Y	
	U	Y	Y	U	Y	Y	U	Y	Y	
	V	Y	Y	V	Y	Y	V	Y	Y	

Figure 32. Frame format in YUV420 data

7.3 RGB888

RGB888 data transmission is performed by transmitting RGB… byte sequence. Each line is separated by line start / end synchronization codes. This sequence is illustrated in Figure 33. The RGB888 frame format is presented in Figure 35.

Bit order in transmission follows the general CCP2 rule, LSB first. This is similar to YUV422 transmission illustrated in Figure 27.

For RGB888 data the suggested way to solve the false synchronization code issue is to constrain the numerical range of R, G and B values from 1 to 254.

Line start	FF	00	00	00	B1	G1	R1	B2	G2	R2	B3	G3
						. . .						
Line end	G638	R638	B639	G639	R639	B640	G640	R640	FF	00	00	01

Figure 33. RGB888 transmission in VGA image case

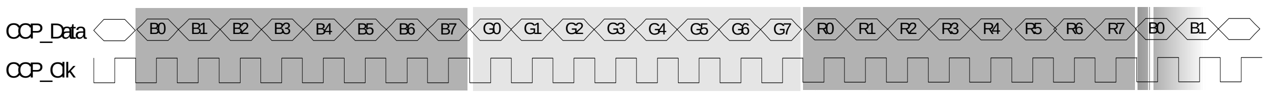


Figure 34. RGB888 transmission in CCP2 bus, bitwise illustration

SOF	B	G	R	B	G	R	B	G	R	SOF
	B	G	R	B	G	R	B	G	R	
	B	G	R	B	G	R	B	G	R	
	B	G	R	B	G	R	B	G	R	
	B	G	R	B	G	R	B	G	R	
SOL	B	G	R	B	G	R	B	G	R	SOL
	B	G	R	B	G	R	B	G	R	
	B	G	R	B	G	R	B	G	R	
	B	G	R	B	G	R	B	G	R	
	B	G	R	B	G	R	B	G	R	

Figure 35. RGB888 Frame format

7.4 RGB565

RGB565 data transmission is performed by transmitting B0..4 G0..5 R0..4 (16-bit) sequence. Each line is separated by line start / end synchronization codes. This sequence is illustrated in Figure 36. The frame format for RGB565 is presented in the Figure 38.
Bit order in transmission follows the general CCP2 rule, LSB first. In RGB565 case the length of one dataword is 16-bits, not 8 bits. The bytewise flip is done for 16-bit BGR words i.e. instead of flipping each byte (8-bits), each 2 bytes (16-bits) are flipped. This is illustrated in Figure 37.
For RGB565 data the suggested way to solve the false synchronization code issue is to constrain the numerical range of G component from 1-63 and R component from 1-31.

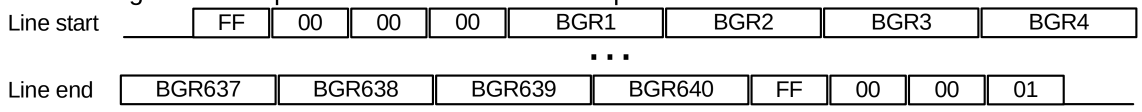


Figure 36. RGB565 transmission in VGA case, note 16-bit length of the BGR-words

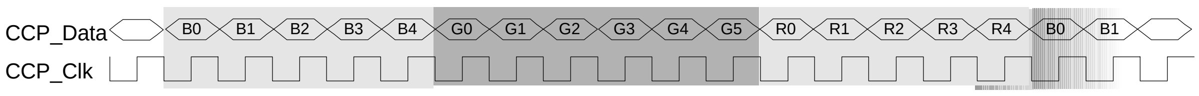


Figure 37. RGB565 transmission on CCP2 bus, bitwise illustration

SOF	BGR	BGR	BGR	BGR	...	BGR	BGR	BGR	SOF
	BGR	BGR	BGR	...	...	...	BGR	BGR	
	BGR	BGR	...	...	...	...	...	BGR	
	BGR	...	...	...	...	...	...	...	
	...	...	...	...	...	...	...	...	
SOL	...	...	...	...	...	...	...	...	SOL
	...	...	...	...	...	...	...	...	
	BGR	...	...	...	...	...	...	BGR	
	BGR	BGR	...	...	...	...	BGR	BGR	
	BGR	BGR	BGR	...	...	BGR	BGR	BGR	

Figure 38. RGB565 Frame format

7.5 RGB444

RGB444 data can be transmitted over CCP2 bus with some special arrangements. The RGB444 data should be made to look like RGB565 data. This can be accomplished by inserting padding bits to LSBs of each color component as illustrated in Figure 40. The padding bits are chosen in a way that false synchronization code cannot appear. Each line is separated by line start / end synchronization codes. This sequence is illustrated in Figure 39. The frame format for RGB444 is presented in the Figure 41.
Bit order in transmission follows the general CCP2 rule, LSB first. In RGB444 case the length of one dataword is 16-bits, not 8 bits. The bytewise flip is done for 16-bit BGR words i.e. instead of flipping each byte (8-bits), each 2 bytes (16-bits) are flipped. This is illustrated in Figure 40.
For RGB444 data the suggested way to solve the false synchronization code issue is to use the padding bits in the LSBs.

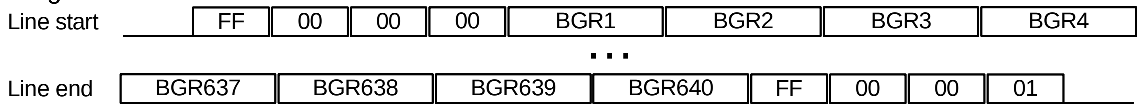


Figure 39. RGB444 transmission in VGA case, note 16-bit length of the BGR-words

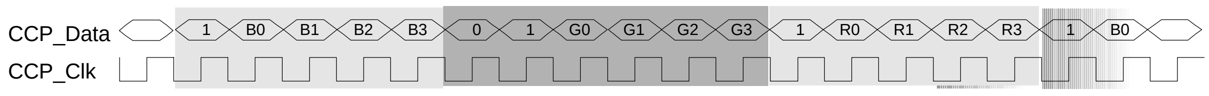


Figure 40. RGB444 transmission on CCP2 bus, bitwise illustration

SOF	BGR	BGR	BGR	BGR	...	BGR	BGR	BGR	SOF
	BGR	BGR	BGR	...	...	...	BGR	BGR	
	BGR	BGR	...	...	...	...	...	BGR	
	BGR	...	...	...	...	...	...	...	
	...	...	...	...	...	...	...	...	
SOL	...	...	...	...	...	...	...	...	SOL
	...	...	...	...	...	...	...	...	
	BGR	...	...	...	...	...	...	BGR	
	BGR	BGR	...	...	...	...	BGR	BGR	
	BGR	BGR	BGR	...	...	BGR	BGR	BGR	

Figure 41. RGB444 frame format

7.6 RAW6

The RAW 6/7/8/10/12 modes are used for transmitting raw Bayer image from the image sensor.

It is possible to transmit e.g. light shielded pixels in addition to effective pixels. This leads to a situation where line length is longer than sum of effective pixels per line. The line length, if not specified otherwise, has to be a multiple of word (32 bits).

The 6-bit Raw Bayer data transmission is performed by transmitting the pixel data over CCP2 bus. Each line is separated by line start / end synchronization codes. This sequence is illustrated in Figure 42 (VGA case). If the number of bits within one line is not a multiple of 32, a padding sequence of 010101… must be used to fulfill the 32-bit requirement.
Each 6-bit pixel is sent LSB first. This is an exception to general CCP2 rule bytewise LSB first.
For 6-bit Raw Bayer the suggested way to solve the false synchronization issue is to constrain the numerical range of pixel values from 1 to 63 (inclusive).

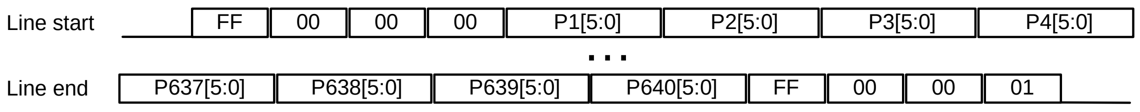


Figure 42. RAW6 data format in VGA case

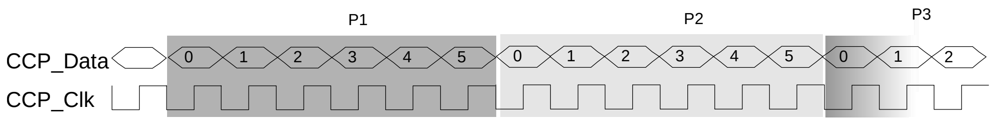
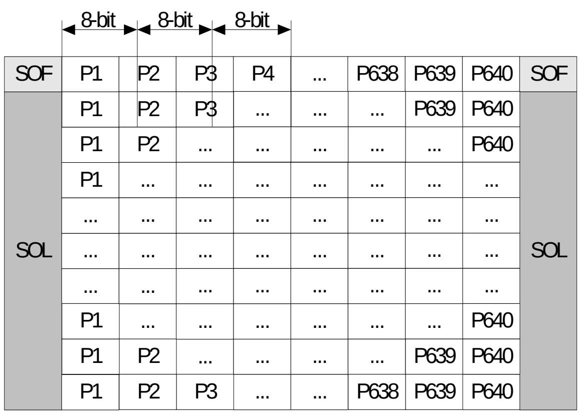


Figure 43. RAW6 data transmission on CCP2 bus, bitwise illustration



SOF	P1	P2	P3	P4	...	P638	P639	P640	SOF
	P1	P2	P3	...	...	...	P639	P640	
	P1	P2	...	...	...	...	...	P640	
	P1	...	...	...	...	...	...	...	
	...	...	...	...	...	...	...	...	
SOL	...	...	...	...	...	...	...	...	SOL
	...	...	...	...	...	...	...	...	
	P1	...	...	...	...	...	...	P640	
	P1	P2	...	...	...	...	P639	P640	
	P1	P2	P3	...	...	P638	P639	P640	

Figure 44. RAW6 Frame format in VGA case

7.7 RAW7

The 7-bit Raw Bayer data transmission is performed by transmitting the pixel data over CCP2 bus. Each line is separated by line start / end synchronization codes. This sequence is illustrated in Figure 45 (VGA case). If the number of bits within one line is not a multiple of 32, a padding sequence of 010101… must be used to fulfill the 32-bit requirement.
Each 7-bit pixel is sent LSB first. This is an exception to general CCP2 rule bytewise LSB first.
For 8-bit Raw Bayer the suggested way to solve the false synchronization issue is to constrain the numerical range of pixel values from 1 to 127 (inclusive).

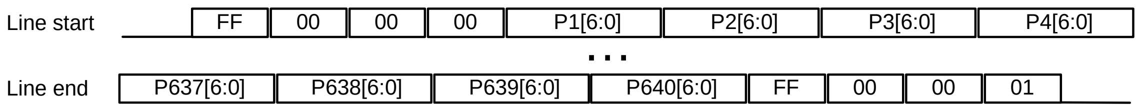


Figure 45. RAW7 data format in VGA case

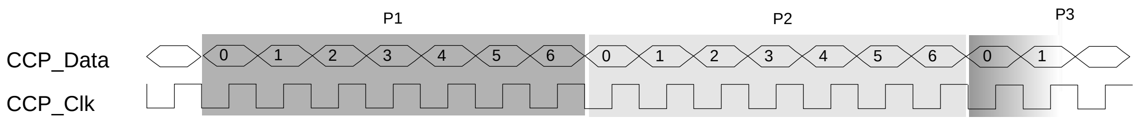
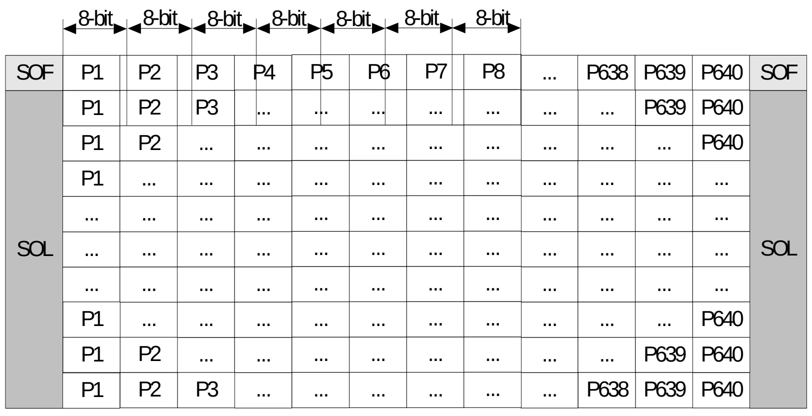


Figure 46. RAW7 data transmission on CCP2 bus, bitwise illustration



SOF	P1	P2	P3	P4	P5	P6	P7	P8	...	P638	P639	P640	SOF
SOL	P1	P2	P3	...	...	...	...	...	...	...	P639	P640	SOL
	P1	P2	...	...	...	...	...	...	...	...	...	P640	
	P1	...	...	...	...	...	...	...	...	...	...	...	
	...	...	...	...	...	...	...	...	...	...	...	...	
	...	...	...	...	...	...	...	...	...	...	...	...	
	...	...	...	...	...	...	...	...	...	...	...	...	
	P1	...	...	...	...	...	...	...	...	...	...	P640	
	P1	P2	...	...	...	...	...	...	...	...	P639	P640	
	P1	P2	P3	...	...	...	...	...	...	P638	P639	P640	

Figure 47. RAW7 Frame format in VGA case

7.8 RAW8

The 8-bit Raw Bayer data transmission is performed by transmitting the pixel data over CCP2 bus. However, the number of pixels between synchronization codes has to be a multiple of 4 pixels or a multiple of 4 bytes. Each line is separated by line start / end synchronization codes. This sequence is illustrated in Figure 48 (VGA case).
Bit order in transmission follows the general CCP2 rule, LSB first.
For 8-bit Raw Bayer the suggested way to solve the false synchronization issue is to constrain the numerical range of pixel values from 1 to 255 (inclusive).

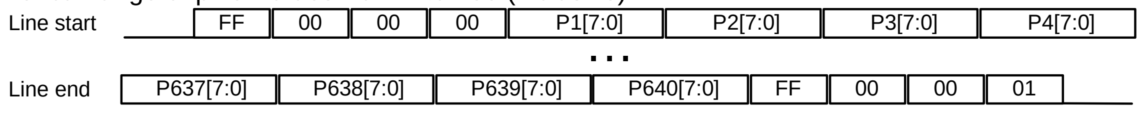


Figure 48. RAW8 data format in VGA case

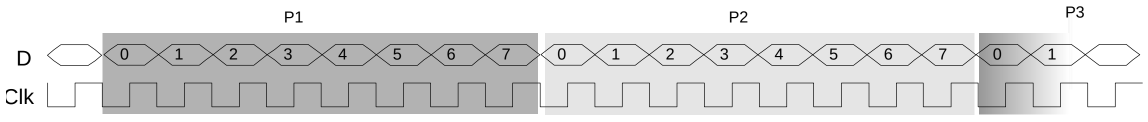


Figure 49. RAW8 data transmission on CCP2 bus, bitwise illustration

SOF	P1	P2	P3	...	P639	P640	EOL
SOL	P1	P2	...	...	...	P640	
	P1	P2	...	...	...	P640	
	P1	...	...	...	...	P640	
	P1	...	...	...	...	P640	
	P1	...	...	...	...	P640	
	P1	...	...	...	...	P640	
	P1	...	...	...	...	P640	
	P1	...	...	...	...	P640	
	P1	...	...	...	...	P640	EOF

Figure 50. RAW8 Frame format in VGA case

7.9 RAW10

The transmission of 10-bit Raw Bayer data is accomplished by packing the 10-bit pixel data to look like 8-bit data format. The number of pixels between synchronization codes has to be a multiple of 16 or a multiple of 20 bytes. Each line is separated by line start / end synchronization codes. This sequence is illustrated in Figure 51 (VGA case).
Bit order in transmission follows the general CCP2 rule, LSB first.
For 10-bit Raw Bayer the suggested way to solve the false synchronization issue is to constrain the numerical range of pixel values from 4 to 1023 (inclusive). Also the every 5th byte (containing LSBs) has to be examined and force the bit 4 (P3 bit 0) ‘1’ if the byte is all zeros.

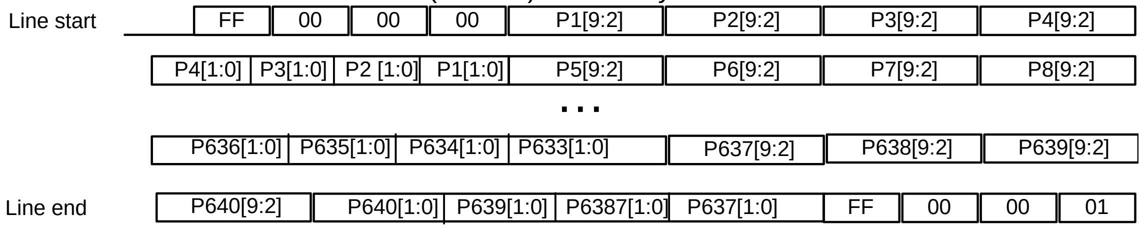


Figure 51. RAW10 data format in VGA case

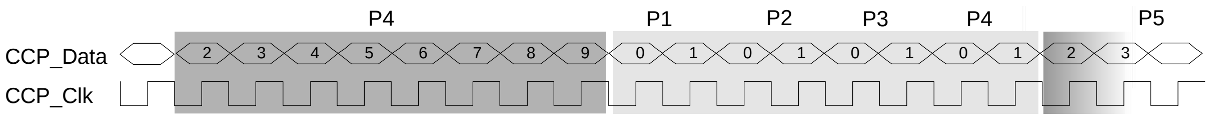


Figure 52. RAW10 data transmission on CCP2 bus, bitwise illustration

SOF	P1	P2	P3	P4	LSBs	...	P640	LSBs	SOF
SOL	P1	P2	P3	...	...	...	P640	LSBs	SOL
	P1	P2	...	...	...	...	P640	LSBs	
	P1	...	...	...	...	...	P640	LSBs	
	...	...	...	...	...	...	P640	LSBs	
	...	...	...	...	...	...	P640	LSBs	
	...	...	...	...	...	...	P640	LSBs	
	P1	...	...	...	...	...	P640	LSBs	
	P1	P2	...	...	...	...	P640	LSBs	
	P1	P2	P3	...	...	P639	P640	LSBs	

Figure 53. RAW10 Frame format in VGA case

7.10 RAW12

The transmission of 12-bit Raw Bayer data is also accomplished by packing the 12-bit pixel data to look like 8-bit data format. The number of pixels between synchronization codes has to be a multiple of 8 pixels or a multiple of 12 bytes. Each line is separated by line start / end synchronization codes. This sequence is illustrated in Figure 54 (VGA case).
Bit order in transmission follows the general CCP2 rule, LSB first.
For 12-bit Raw Bayer the suggested way to solve the false synchronization issue is to constrain the numerical range of pixel values from 16 to 4095 (inclusive). Also the every 3rd byte (containing LSBs) has to be examined and force the bit 4 (P2 bit 0) '1' if the byte is all zeros.

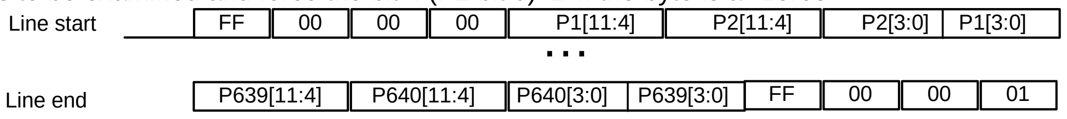


Figure 54. RAW12 data format in VGA case

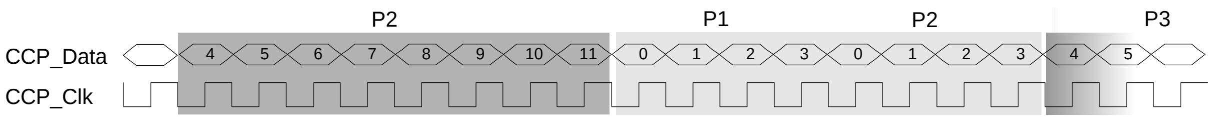


Figure 55. RAW12 transmission on CCP2 bus, bitwise illustration

SOF	P1	P2	LSBs	P3	P4	...	P640	LSBs	SOF
	P1	P2	LSBs	...	...	...	P640	LSBs	
	P1	P2	...	...	...	...	P640	LSBs	
	P1	...	...	...	...	...	P640	LSBs	
	...	...	...	...	...	...	P640	LSBs	
SOL	...	...	...	...	...	...	P640	LSBs	SOL
	...	...	...	...	...	...	P640	LSBs	
	P1	...	...	...	...	...	P640	LSBs	
	P1	P2	...	...	...	P639	P640	LSBs	
	P1	P2	LSBs	...	LSBs	P639	P640	LSBs	

Figure 56. RAW12 frame format in VGA case

7.11 Compressed data formats

It is possible to transmit compressed data formats, such as JPEG, over CCP2 bus with certain restrictions:

- The transmitted data has to be encoded with false synchronization code protection algorithm (FSP). The FSP principle is illustrated in Figure 61
- Only frame synchronization codes are applied. The use of synchronization codes is illustrated in Figure 57
- The transmitted data length in bits has to be divisible by 32, i.e. whole 32 bit words are transmitted. If the transmitted data does not have even amount of bytes, the missing bytes must be padded in the end of the data sequence. A5h value is used for padding.

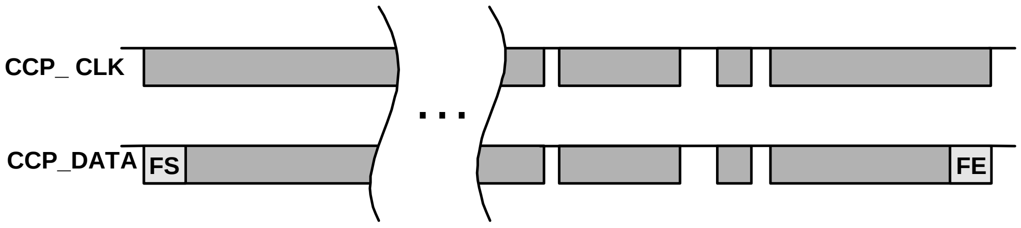


Figure 57. Use of synchronization signals in transmission of compressed data

Figure 57 illustrates the data transfer principle of compressed data format on CCP2. Only Frame start and Frame end synchronization codes are used. This is also shown in Figure 58.
Because compressed data frames are varying in length, flexibility has to be allowed in the transmission. It is possible to simply stop the data transfer if there is no data to be transferred, by switching off the transmission clock. This is shown in Figure 57.

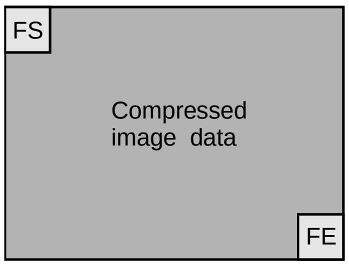


Figure 58. Frame structure with compressed data format

7.11.1 JPEG8

Any non-standard or additional data inside the baseline JPEG data structure has to be removed from JPEG8 data before it is compliant with e.g. standard JPEG image viewers in e.g. a personal computer. Also, in encoding side, the additional information has to be embedded in the data before FSP encoding (refer to chapter 7.11.1.1), and removed after FSP decoding (refer to chapter 7.11.1.2). The JPEG8 data flow is illustrated in the Figure 59 and Figure 60 below.

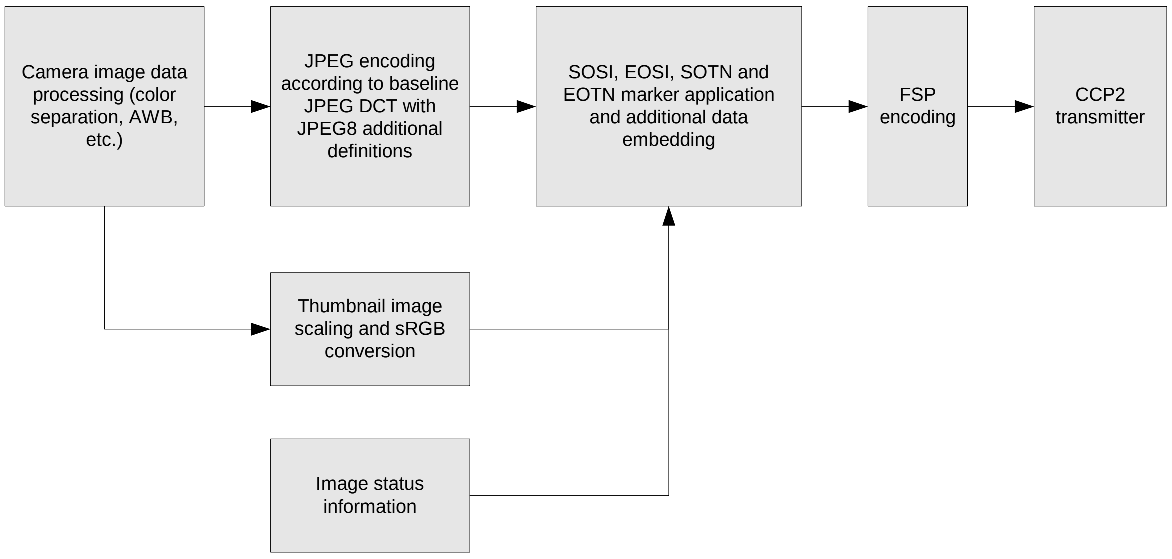

Camera image data processing (color separation, AWB, etc.)
JPEG encoding according to baseline JPEG DCT with JPEG8 additional definitions
SOSI, EOSI, SOTN and EOTN marker application and additional data embedding
FSP encoding
CCP2 transmitter
Thumbnail image scaling and sRGB conversion
Image status information

Figure 59. JPEG8 data flow in encoder

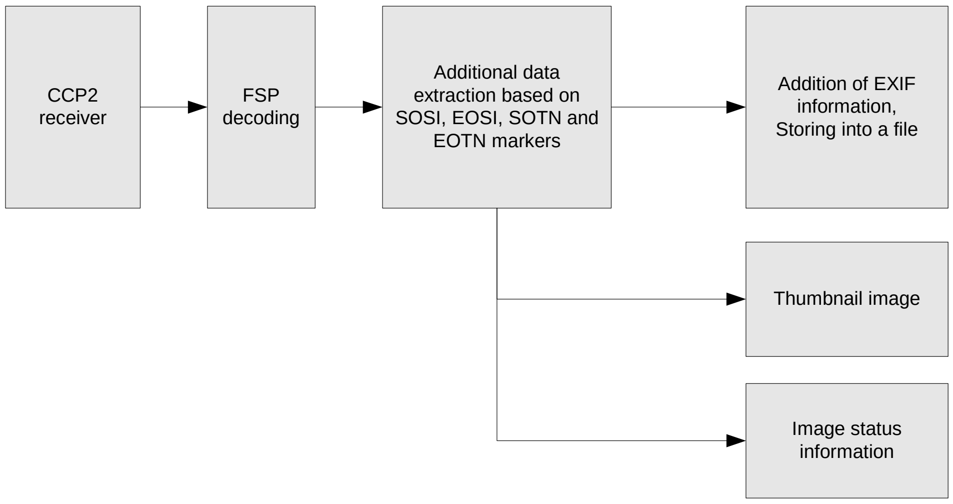

CCP2 receiver
FSP decoding
Additional data extraction based on SOSI, EOSI, SOTN and EOTN markers
Addition of EXIF information, Storing into a file
Thumbnail image
Image status information

Figure 60. JPEG8 data flow in the decoder

7.11.1.1 FSP encoding

The FSP encoding is done for byte aligned data. The principle is to search for 00h bytes, and if found, check if there is any of the sequences listed in Table 10 in the previous 2 bytes, which would cause data sequence FF0000h to appear on CCP2 in bitwise inspection (CCP2 receiver detects synchronization codes bitwise). Because of data flipping to LSB first on CCP2 bus, the table lists both the data appearance on CCP2 and in JPEG byte data.
If an illegal sequence is found, one byte of A5h is added after the detected 00h (byte n in Table 10), i.e. byte n+1 = A5h. Note that A5h (1010 0101 b) remains as A5h even after flipping (1010 0101 flipped = 1010 0101). The flowchart in Figure 61 illustrates the algorithm operation. The 8 violating combinations can be easily searched and found with both HW and SW implementations.
Operation of the FSP encoder and JPEG over CCP2 transmmitter is illustrated in Figure 62.

Illegal bit patterns in the normal, MSB first JPEG byte data			Resulting illegal bit patterns in byte flipped LSB first data transmitted on CCP2		
Byte n-2 In JPEG	Byte n-1 In JPEG	Detected 00h (Byte n) in JPEG	Byte n-2 on CCP2	Byte n-1 on CCP2	Byte n on CCP2
????????	11111111	00000000	????????	11111111	00000000
1???????	01111111	00000000	???????1	11111110	00000000
11??????	00111111	00000000	??????11	11111100	00000000
111?????	00011111	00000000	?????111	11111000	00000000
1111????	00001111	00000000	????1111	11110000	00000000
11111???	00000111	00000000	???11111	11100000	00000000
111111??	00000011	00000000	??111111	11000000	00000000
1111111?	00000001	00000000	?1111111	10000000	00000000

Table 10. FSP detected sequences

The FSP encoder can be implemented in HW by having 3 bytes of memory, and a FIFO buffer which is needed anyway for JPEG data. If a 00h is detected in byte n, combinatorial logic can be used for detecting is there one of the error sequences in bytes n-1 and n-2.

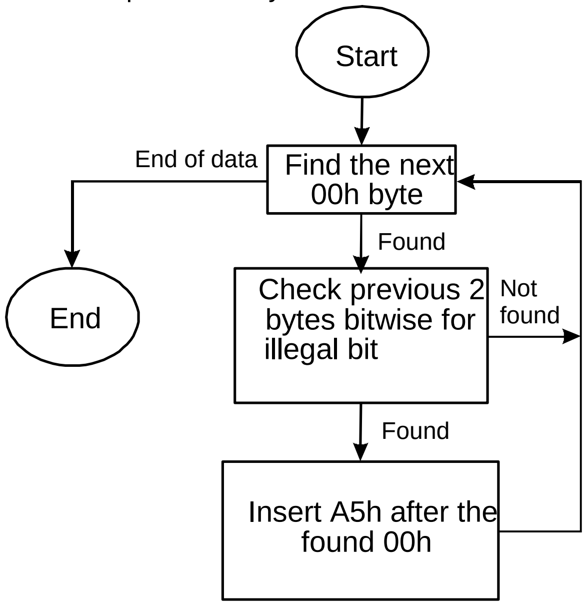

Start
End of data
Find the next 00h byte
Found
End
Check previous 2 bytes bitwise for illegal bit
Not found
Found
Insert A5h after the found 00h

Figure 61. False synchronization code protection encoding algorithm for big endian byte data

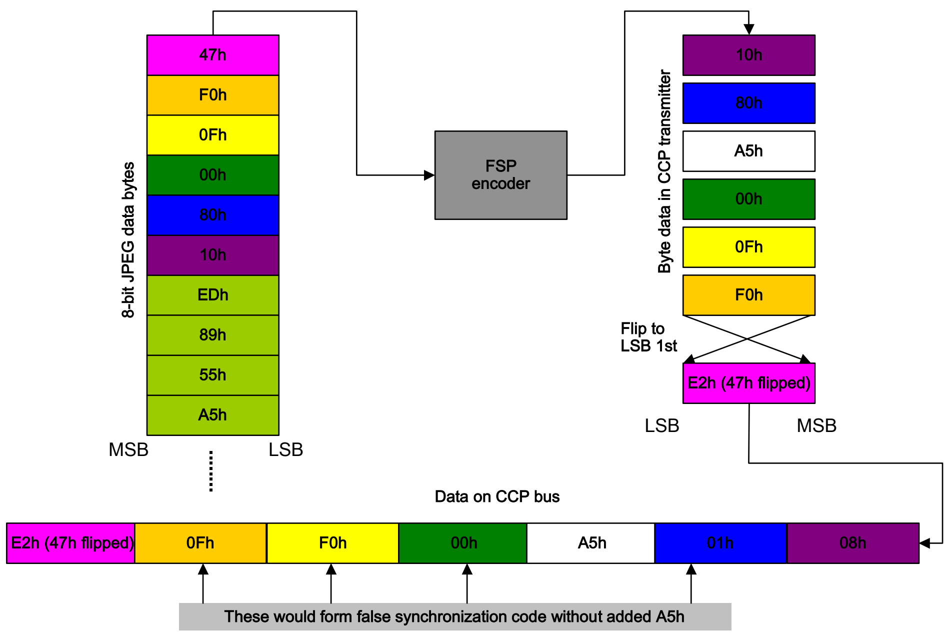


Figure 62. FSP encoder and CCP2 transmitter data flow principle

7.11.1.2 FSP Decoding

The FSP decoder is in principle very similar to the encoder. The idea is to detect the zeros & false sequences (as described in Table 10), and if found, to remove the extra A5h appearing after the error sequence.

JPEG8 data is decoded according to the algorithm presented in Figure 63.

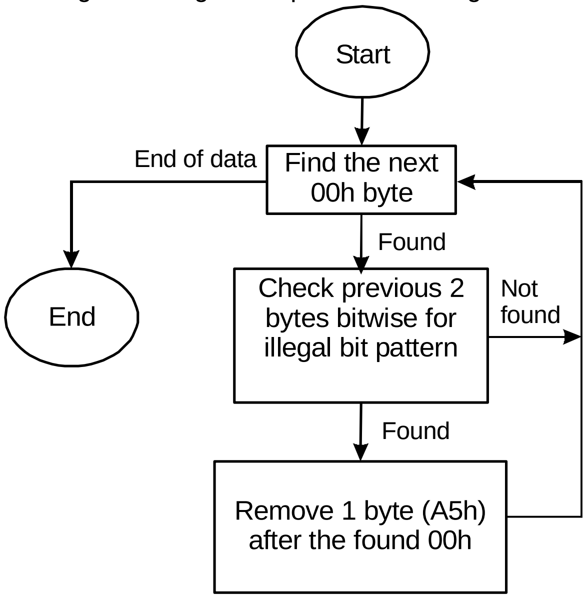

Start
End of data
Find the next 00h byte
Found
End
Check previous 2 bytes bitwise for illegal bit pattern
Not found
Found
Remove 1 byte (A5h) after the found 00h

Figure 63. False synchronization code protection decoding algorithm for byte data

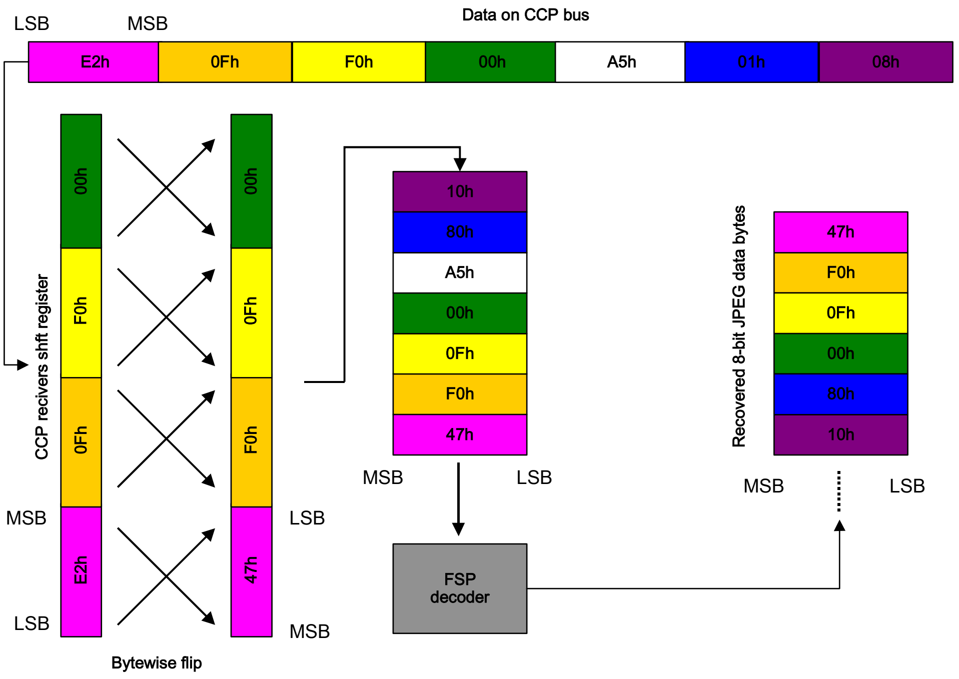


Figure 64. Example FSP decoding data flow

7.11.1.3 JPEG data definition

The JPEG data generated in camera module is baseline JPEG DCT format defined in ISO/IEC 10918-1, with following additional definitions or modifications:

- sRGB color space shall be used. The JPEG is generated from YcbCr format after sRGB to YcbCr conversion.
- The JPEG metadata has to be Exif compatible, i.e. metadata within application segments has to be placed in beginning of file, in order which is illustrated in Figure 65
- Status line is added in the end of JPEG data as defined in chapter 7.11.1.3.1
- If needed, a thumbnail image is interlaced in order which is free of choice as defined in chapter 7.11.1.3.2
- Prior to storing into a file, the CCP2 JPEG data is processed by the data separation process described in the chapter 7.11.

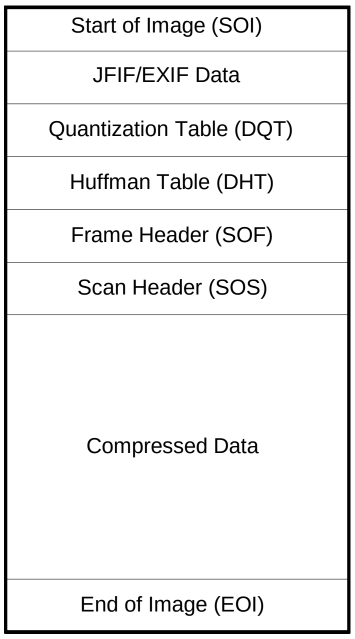


Figure 65. Exif compatible baseline JPEG DCT format

7.11.1.3.1 Image status information

Information of at least the following items has to be stored in the end of the JPEG sequence as illustrated in Figure 66:

1. Image exposure time
2. Analog & digital gains used
3. White balancing gains for each color component
4. Camera version number
5. Camera register settings
6. Image resolution and possible thumbnail resolution

The camera register settings may include a subset of camera’s registers. The essential information needed for JPEG8 image is the information needed for converting the image back to linear space. This is necessary e.g. for printing service. An example of register settings is following:

- Sample frequency
- Exposure
- Analog and digital gain
- Gamma
- Color gamut conversion matrix
- Contrast
- Brightness
- Pre-gain

The status information content has to be defined in the product specification of each camera module containing the JPEG8 feature. The format and content is manufacturer specific.

The image status data should be arranged so that each byte is split into 2 4-bit nibbles and “1010” padding sequence is added to MSB, as presented in the Table 11 below. This ensures that no JPEG escape sequences (0xFF 0x00) are present in the status data.

Data word	After padding
D7D6D5D4D3D2D1D0	1010D7D6D5D4 1010D3D2D1D0

Table 11. Status data padding

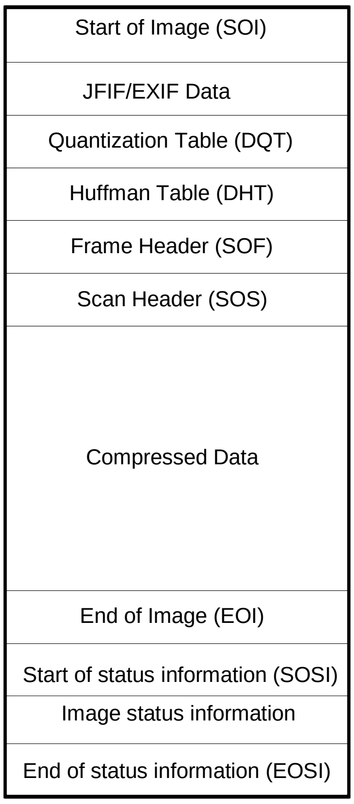


Figure 66. Status information field in the end of baseline JPEG frame

The SOSI and EOSI markers are defined in 7.11.1.3.3.

7.11.1.3.2 Thumbnail image

A thumbnail image may be embedded inside the JPEG data, if needed. That means, the thumbnail feature is not compulsory for each camera module containing the JPEG8 feature.
The philosophy of embedded / interlaced thumbnail addition is to minimise the needed frame memory. The TN (ThumbNail) data can be included in any part of the compressed image data segment (see Figure 67), and in as many pieces as needed
TN data is separated from compressed data by SOTN (Start Of ThumbNail) and EOTN (End Of ThumbNail) non-standard markers, which are defined in 7.11.1.3.3. The amount of fields sparated by SOTN and EOTN is not limited.
The format for the TN data is 24-bit RGB, in R,G,B,R,G…. byte order and sRGB color space. Each TN data field separated by the SOTN / EOTN markers has to contain equal amount of R, G, and B bytes, this is, complete pixels. Thumbnail image resolution will be defined separately in each camera modules product specification.
The thumbnail image data is decoded and removed apart from the JPEG compressed data prior to writing the JPEG into a file. In the process, TN data fields are appended one after each other, in order of occurrence in the received JPEG data.

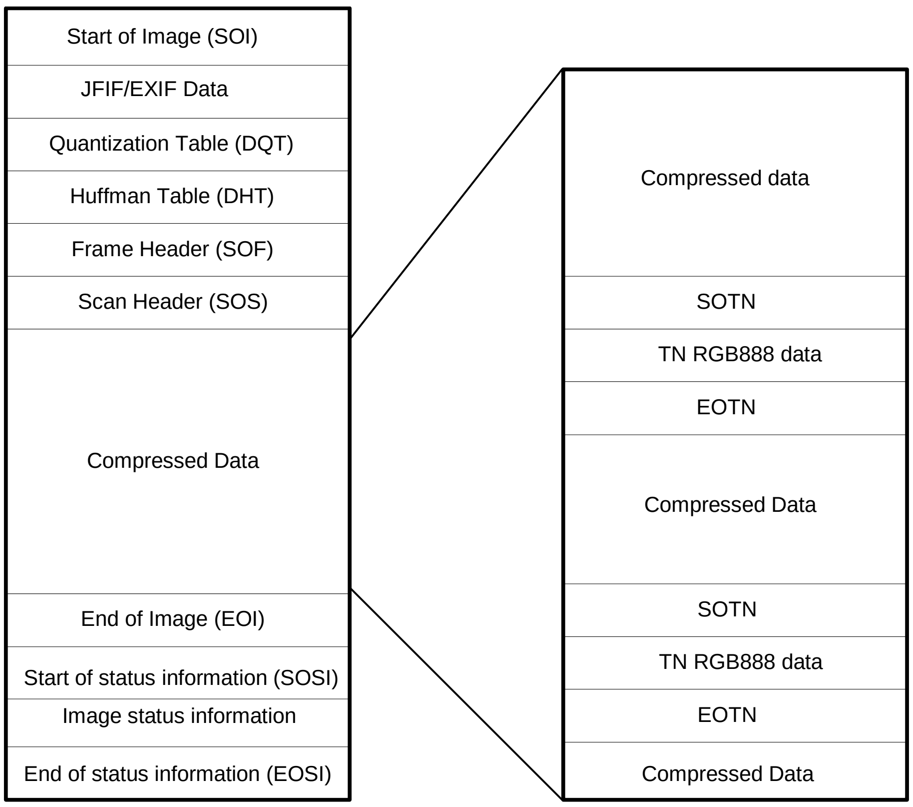


Figure 67. Example of TN image embedding inside the compressed JPEG data block

7.11.1.3.3 JPEG8 non-standard markers

JPEG8 uses the reserved JPEG data markers for special purposes, marking the additional segments inside the data file. These segments are not part of the JPEG, JFIF [**1**], EXIF [**2**] or any other specifications, instead their use is specified in this document in chapters 7.11.1.3.1 and 7.11.1.3.2.
The use of the non-standard markers is always internal to a product containing the JPEG8 camera module, and these markers are always removed from the JPEG data before storing it into a file.

Non-standard Marker symbol	Marker data code used
SOSI	FFh BCh
EOSI	FFh BDh
SOTN	FFh BEh
EOTN	FFh BFh

Table 12. Listing of JPEG8 additional marker codes

8. Recommended CCP2 receiver behavior

The reception of the image data is based on synchronization codes. After initialization the CCP2 receiver should start looking for frame start synchronization code and synchronize itself to the incoming data stream. The receiver should also keep track on synchronization codes so that if false synchronization code occurs or synchronization sequence has been shifted the receiver should inform the host system.

The false synchronization code error means that wrong synchronization code has been detected in the receiver, e.g. after SOL synchronization code a new SOL code is detected instead of EOL synchronization code.

Shifted synchronization code error means that correct synchronization code has been detected, but the alignment of the synchronization code is not correct. For example due to noise coupled in the transmission media additional change in the data or strobe line has been generated thus creating an extra clock edge in the receiver.

If false synchronization code occurs the receiver can expect the host system to handle that, but in the case of shifted synchronization code, the receiver should resynchronize itself to the next synchronization code. This is illustrated in the example state diagram below.

In a case of checksum mismatch, the receiver should inform the host system, but the receiver should continue receiving the data, unless the host system instructs to do otherwise.

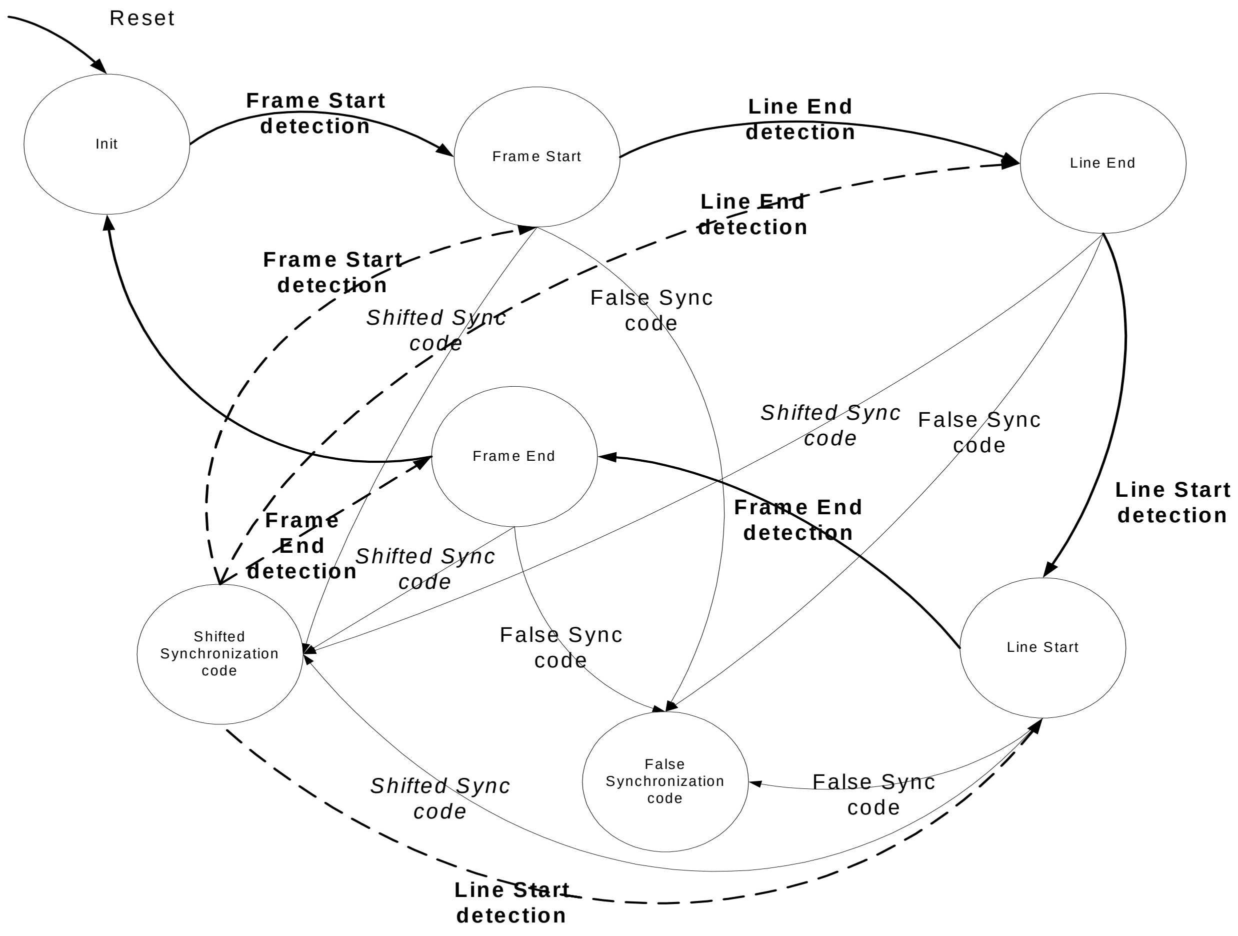


Figure 68. An example state diagram for receiver behavior

The transmitter can use free running or gated clock to create the data and strobe/clock signals. If gated transmission clock is used, the receiver implementation has to be able to react to the frame end synchronization codes immediately. This is necessary to prevent the dropping of frames in the receiver.
The CCP2 compatible receiver should be able to operate with data-strobe signaling or data-clock signaling.
The CCP2 data protocol requires certain behavior from the receiver connected to the CCP2 transmitter. In the following figures there is explained how different data formats should be handled inside the receiver

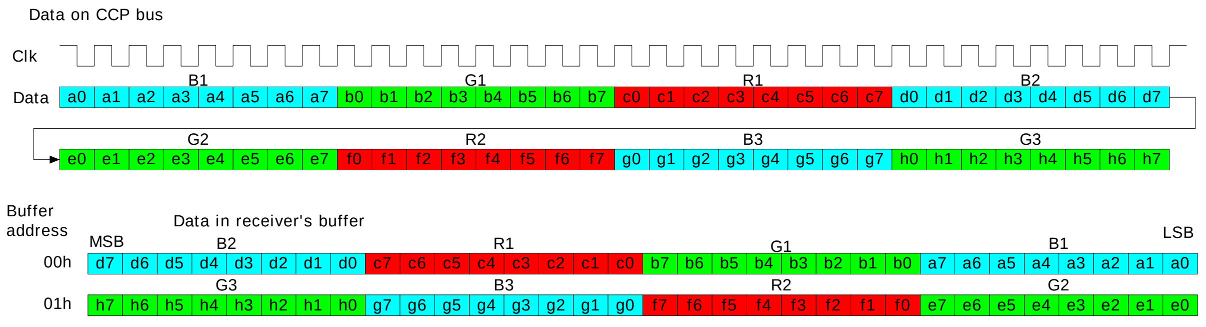

Data on CCP bus
Clk
Data
B1 G1 R1 B2
a0 a1 a2 a3 a4 a5 a6 a7 b0 b1 b2 b3 b4 b5 b6 b7 c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7
G2 R2 B3 G3
e0 e1 e2 e3 e4 e5 e6 e7 f0 f1 f2 f3 f4 f5 f6 f7 g0 g1 g2 g3 g4 g5 g6 g7 h0 h1 h2 h3 h4 h5 h6 h7
Buffer address
Data in receiver's buffer
MSB
LSB
B2 R1 G1 B1
00h
d7 d6 d5 d4 d3 d2 d1 d0 c7 c6 c5 c4 c3 c2 c1 c0 b7 b6 b5 b4 b3 b2 b1 b0 a7 a6 a5 a4 a3 a2 a1 a0
G3 B3 R2 G2
01h
h7 h6 h5 h4 h3 h2 h1 h0 g7 g6 g5 g4 g3 g2 g1 g0 f7 f6 f5 f4 f3 f2 f1 f0 e7 e6 e5 e4 e3 e2 e1 e0

Figure 69. The RGB888 data format reception

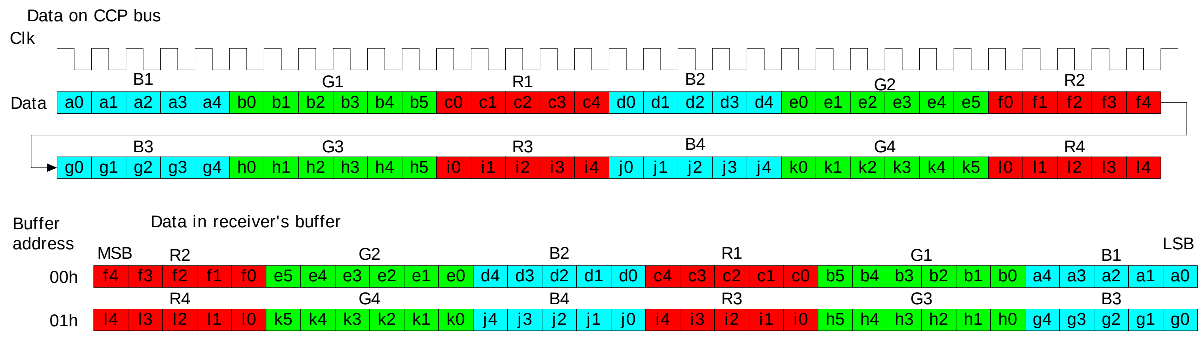

Data on CCP bus
Clk
Data
B1 G1 R1 B2 G2 R2
a0 a1 a2 a3 a4 b0 b1 b2 b3 b4 b5 c0 c1 c2 c3 c4 d0 d1 d2 d3 d4 e0 e1 e2 e3 e4 e5 f0 f1 f2 f3 f4
B3 G3 R3 B4 G4 R4
g0 g1 g2 g3 g4 h0 h1 h2 h3 h4 h5 i0 i1 i2 i3 i4 j0 j1 j2 j3 j4 k0 k1 k2 k3 k4 k5 l0 l1 l2 l3 l4
Buffer address
Data in receiver's buffer
MSB
LSB
R2 G2 B2 R1 G1 B1
00h
f4 f3 f2 f1 f0 e5 e4 e3 e2 e1 e0 d4 d3 d2 d1 d0 c4 c3 c2 c1 c0 b5 b4 b3 b2 b1 b0 a4 a3 a2 a1 a0
R4 G4 B4 R3 G3 B3
01h
l4 l3 l2 l1 l0 k5 k4 k3 k2 k1 k0 j4 j3 j2 j1 j0 i4 i3 i2 i1 i0 h5 h4 h3 h2 h1 h0 g4 g3 g2 g1 g0

Figure 70. The RGB565 data format reception

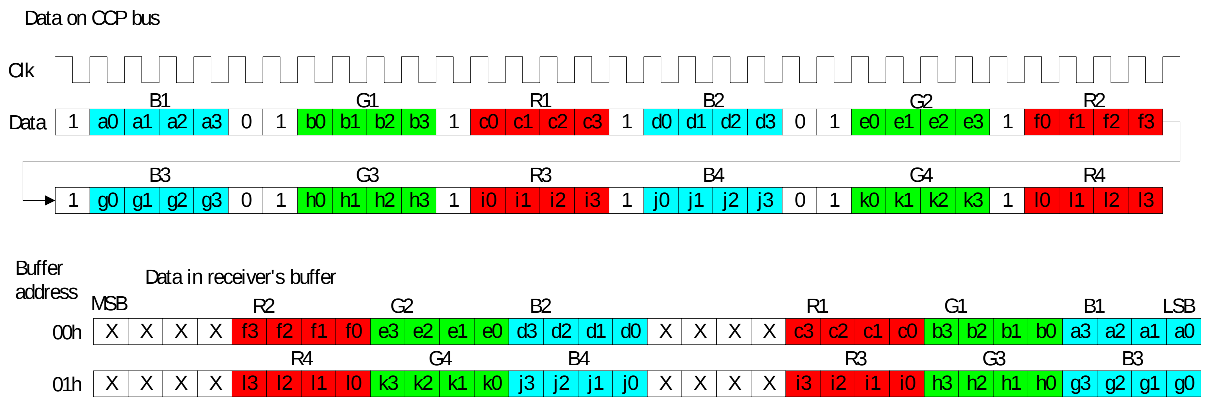

Data on CCP bus
Clk
Data
B1 G1 R1 B2 G2 R2
1 a0 a1 a2 a3 0 1 b0 b1 b2 b3 1 c0 c1 c2 c3 1 d0 d1 d2 d3 0 1 e0 e1 e2 e3 1 f0 f1 f2 f3
B3 G3 R3 B4 G4 R4
1 g0 g1 g2 g3 0 1 h0 h1 h2 h3 1 i0 i1 i2 i3 1 j0 j1 j2 j3 0 1 k0 k1 k2 k3 1 l0 l1 l2 l3
Buffer address
Data in receiver's buffer
MSB
LSB
R2 G2 B2 R1 G1 B1
00h
X X X X f3 f2 f1 f0 e3 e2 e1 e0 d3 d2 d1 d0 X X X X c3 c2 c1 c0 b3 b2 b1 b0 a3 a2 a1 a0
R4 G4 B4 R3 G3 B3
01h
X X X X l3 l2 l1 l0 k3 k2 k1 k0 j3 j2 j1 j0 X X X X i3 i2 i1 i0 h3 h2 h1 h0 g3 g2 g1 g0

Figure 71. The RGB444 data format reception

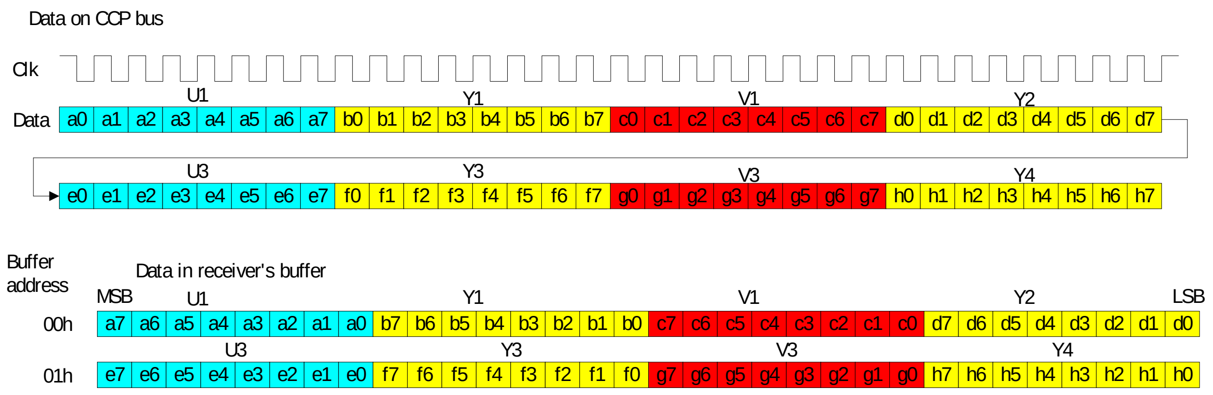

Data on CCP bus
Clk
Data
U1 Y1 V1 Y2
a0 a1 a2 a3 a4 a5 a6 a7 b0 b1 b2 b3 b4 b5 b6 b7 c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7
U3 Y3 V3 Y4
e0 e1 e2 e3 e4 e5 e6 e7 f0 f1 f2 f3 f4 f5 f6 f7 g0 g1 g2 g3 g4 g5 g6 g7 h0 h1 h2 h3 h4 h5 h6 h7
Buffer address
Data in receiver's buffer
MSB
LSB
00h
a7 a6 a5 a4 a3 a2 a1 a0 b7 b6 b5 b4 b3 b2 b1 b0 c7 c6 c5 c4 c3 c2 c1 c0 d7 d6 d5 d4 d3 d2 d1 d0
01h
e7 e6 e5 e4 e3 e2 e1 e0 f7 f6 f5 f4 f3 f2 f1 f0 g7 g6 g5 g4 g3 g2 g1 g0 h7 h6 h5 h4 h3 h2 h1 h0

Figure 72. The YUV422 data format reception

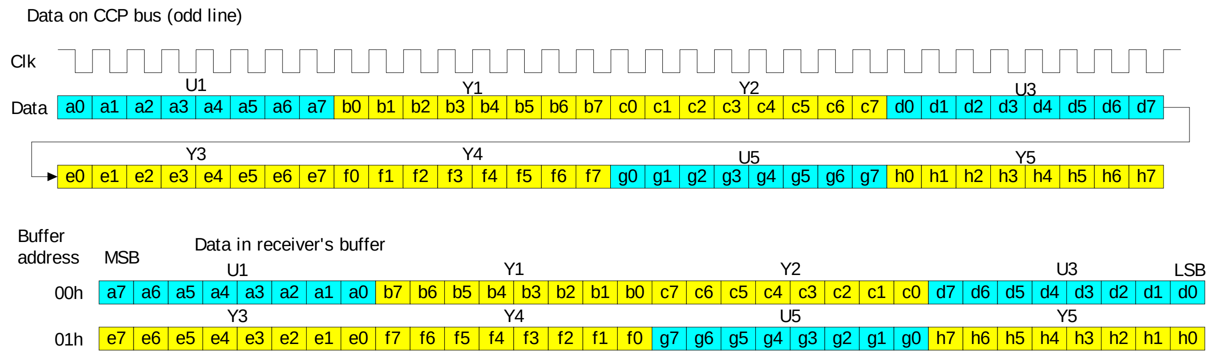

Data on CCP bus (odd line)
Clk
Data
U1 Y1 Y2 U3
a0 a1 a2 a3 a4 a5 a6 a7 b0 b1 b2 b3 b4 b5 b6 b7 c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7
Y3 Y4 U5 Y5
e0 e1 e2 e3 e4 e5 e6 e7 f0 f1 f2 f3 f4 f5 f6 f7 g0 g1 g2 g3 g4 g5 g6 g7 h0 h1 h2 h3 h4 h5 h6 h7
Buffer address
MSB
Data in receiver's buffer
LSB
00h
a7 a6 a5 a4 a3 a2 a1 a0 b7 b6 b5 b4 b3 b2 b1 b0 c7 c6 c5 c4 c3 c2 c1 c0 d7 d6 d5 d4 d3 d2 d1 d0
01h
e7 e6 e5 e4 e3 e2 e1 e0 f7 f6 f5 f4 f3 f2 f1 f0 g7 g6 g5 g4 g3 g2 g1 g0 h7 h6 h5 h4 h3 h2 h1 h0

Figure 73. The YUV420data format reception

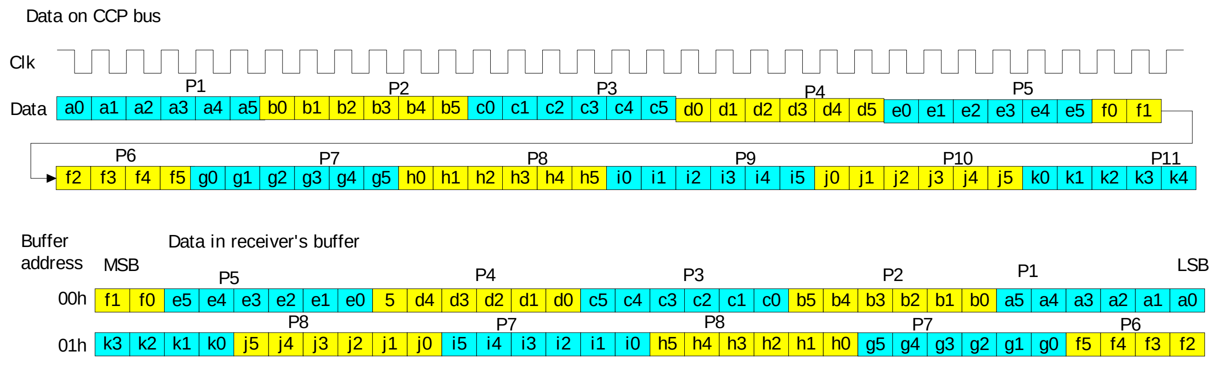

Data on CCP bus
Clk
Data
P1 P2 P3 P4 P5
a0 a1 a2 a3 a4 a5 b0 b1 b2 b3 b4 b5 c0 c1 c2 c3 c4 c5 d0 d1 d2 d3 d4 d5 e0 e1 e2 e3 e4 e5 f0 f1
P6 P7 P8 P9 P10 P11
f2 f3 f4 f5 g0 g1 g2 g3 g4 g5 h0 h1 h2 h3 h4 h5 i0 i1 i2 i3 i4 i5 j0 j1 j2 j3 j4 j5 k0 k1 k2 k3 k4
Buffer address
MSB
Data in receiver's buffer
LSB
P5 P4 P3 P2 P1
00h
f1 f0 e5 e4 e3 e2 e1 e0 5 d4 d3 d2 d1 d0 c5 c4 c3 c2 c1 c0 b5 b4 b3 b2 b1 b0 a5 a4 a3 a2 a1 a0
P8 P7 P8 P7 P6
01h
k3 k2 k1 k0 j5 j4 j3 j2 j1 j0 i5 i4 i3 i2 i1 i0 h5 h4 h3 h2 h1 h0 g5 g4 g3 g2 g1 g0 f5 f4 f3 f2

Figure 74. The RAW6 data format reception

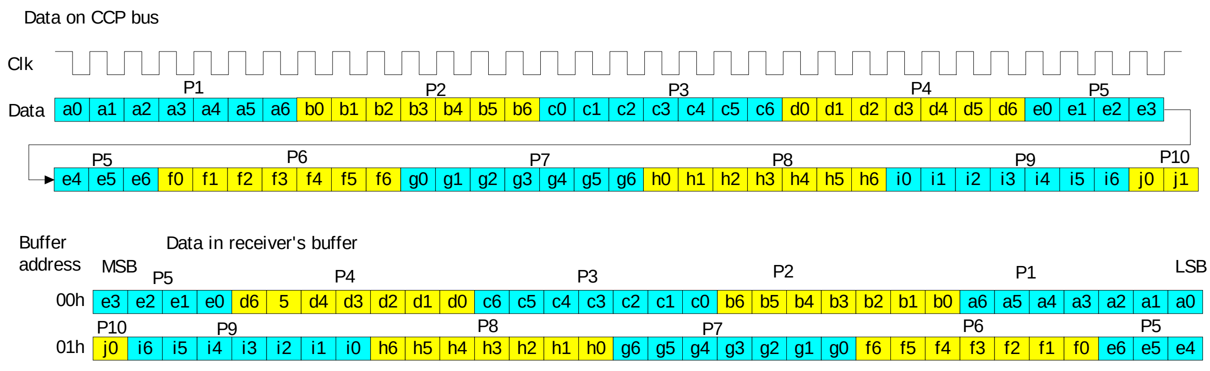

Data on CCP bus
Clk
Data
P1 P2 P3 P4 P5
a0 a1 a2 a3 a4 a5 a6 b0 b1 b2 b3 b4 b5 b6 c0 c1 c2 c3 c4 c5 c6 d0 d1 d2 d3 d4 d5 d6 e0 e1 e2 e3
P5 P6 P7 P8 P9 P10
e4 e5 e6 f0 f1 f2 f3 f4 f5 f6 g0 g1 g2 g3 g4 g5 g6 h0 h1 h2 h3 h4 h5 h6 i0 i1 i2 i3 i4 i5 i6 j0 j1
Buffer address
Data in receiver's buffer
MSB
LSB
P5 P4 P3 P2 P1
00h
e3 e2 e1 e0 d6 5 d4 d3 d2 d1 d0 c6 c5 c4 c3 c2 c1 c0 b6 b5 b4 b3 b2 b1 b0 a6 a5 a4 a3 a2 a1 a0
P10 P9 P8 P7 P6 P5
01h
j0 i6 i5 i4 i3 i2 i1 i0 h6 h5 h4 h3 h2 h1 h0 g6 g5 g4 g3 g2 g1 g0 f6 f5 f4 f3 f2 f1 f0 e6 e5 e4

Figure 75. The RAW7 data format reception

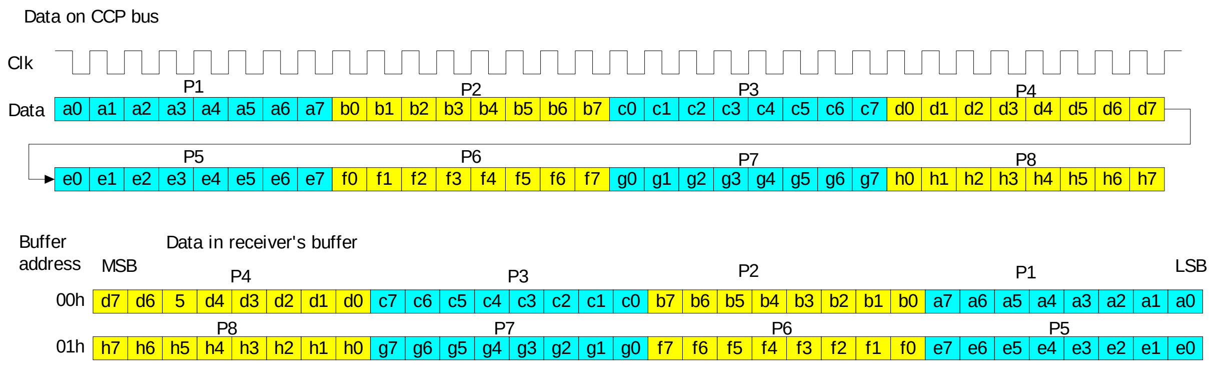

Data on CCP bus
Clk
Data
P1 P2 P3 P4
a0 a1 a2 a3 a4 a5 a6 a7 b0 b1 b2 b3 b4 b5 b6 b7 c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7
P5 P6 P7 P8
e0 e1 e2 e3 e4 e5 e6 e7 f0 f1 f2 f3 f4 f5 f6 f7 g0 g1 g2 g3 g4 g5 g6 g7 h0 h1 h2 h3 h4 h5 h6 h7
Buffer address
Data in receiver's buffer
MSB
LSB
P4 P3 P2 P1
00h
d7 d6 5 d4 d3 d2 d1 d0 c7 c6 c5 c4 c3 c2 c1 c0 b7 b6 b5 b4 b3 b2 b1 b0 a7 a6 a5 a4 a3 a2 a1 a0
P8 P7 P6 P5
01h
h7 h6 h5 h4 h3 h2 h1 h0 g7 g6 g5 g4 g3 g2 g1 g0 f7 f6 f5 f4 f3 f2 f1 f0 e7 e6 e5 e4 e3 e2 e1 e0

Figure 76. The RAW8 data format reception

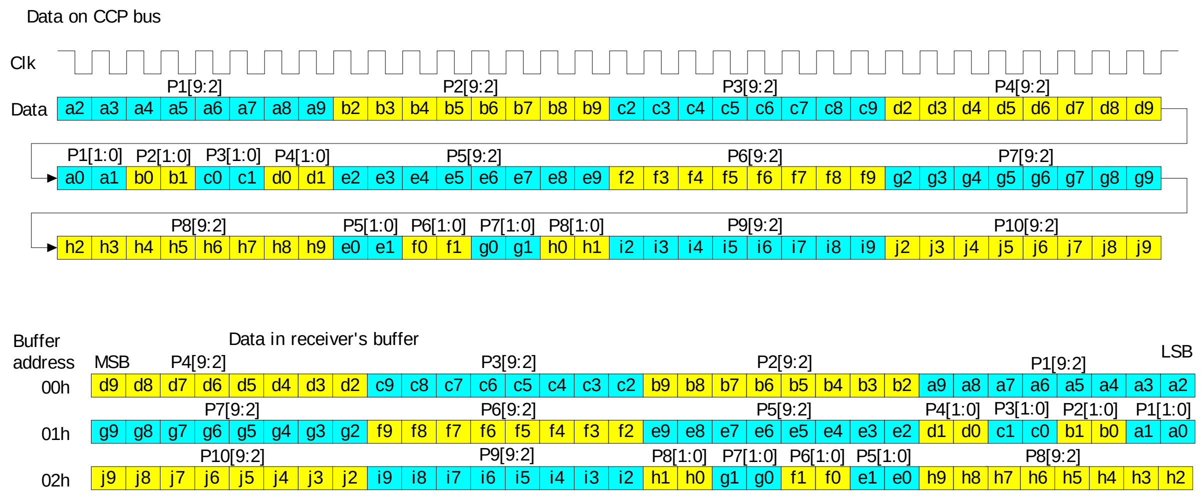

Data on CCP bus
Clk
Data
P1[9:2] P2[9:2] P3[9:2] P4[9:2]
a2 a3 a4 a5 a6 a7 a8 a9 b2 b3 b4 b5 b6 b7 b8 b9 c2 c3 c4 c5 c6 c7 c8 c9 d2 d3 d4 d5 d6 d7 d8 d9
P1[1:0] P2[1:0] P3[1:0] P4[1:0] P5[9:2] P6[9:2] P7[9:2]
a0 a1 b0 b1 c0 c1 d0 d1 e2 e3 e4 e5 e6 e7 e8 e9 f2 f3 f4 f5 f6 f7 f8 f9 g2 g3 g4 g5 g6 g7 g8 g9
P8[9:2] P5[1:0] P6[1:0] P7[1:0] P8[1:0] P9[9:2] P10[9:2]
h2 h3 h4 h5 h6 h7 h8 h9 e0 e1 f0 f1 g0 g1 h0 h1 i2 i3 i4 i5 i6 i7 i8 i9 j2 j3 j4 j5 j6 j7 j8 j9
Buffer address
Data in receiver's buffer
MSB
LSB
P4[9:2] P3[9:2] P2[9:2] P1[9:2]
00h
d9 d8 d7 d6 d5 d4 d3 d2 c9 c8 c7 c6 c5 c4 c3 c2 b9 b8 b7 b6 b5 b4 b3 b2 a9 a8 a7 a6 a5 a4 a3 a2
P7[9:2] P6[9:2] P5[9:2] P4[1:0] P3[1:0] P2[1:0] P1[1:0]
01h
g9 g8 g7 g6 g5 g4 g3 g2 f9 f8 f7 f6 f5 f4 f3 f2 e9 e8 e7 e6 e5 e4 e3 e2 d1 d0 c1 c0 b1 b0 a1 a0
P10[9:2] P9[9:2] P8[1:0] P7[1:0] P6[1:0] P5[1:0] P8[9:2]
02h
j9 j8 j7 j6 j5 j4 j3 j2 i9 i8 i7 i6 i5 i4 i3 i2 h1 h0 g1 g0 f1 f0 e1 e0 h9 h8 h7 h6 h5 h4 h3 h2

Figure 77. The RAW10 data format reception

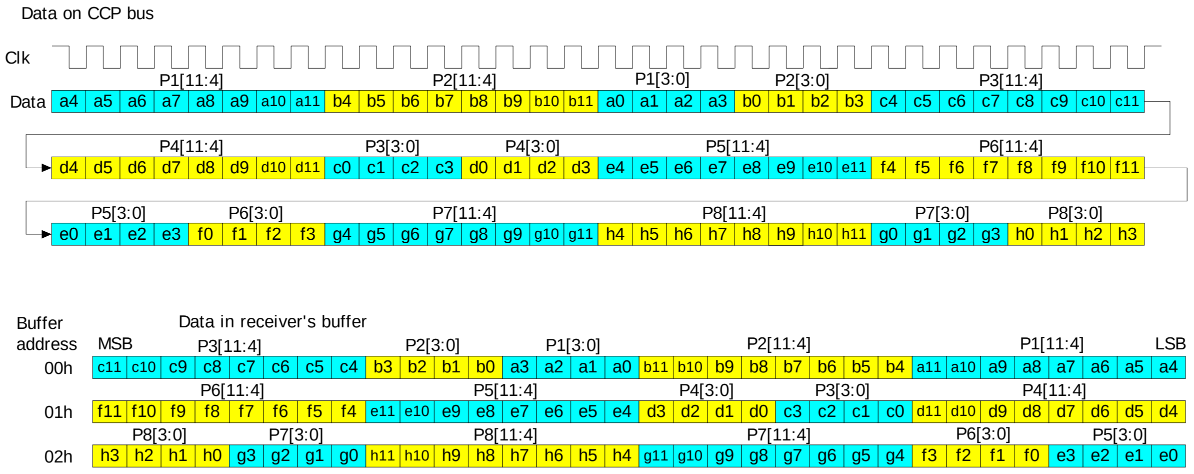

Data on CCP bus
Clk
Data
P1[11:4] P2[11:4] P1[3:0] P2[3:0] P3[11:4]
a4 a5 a6 a7 a8 a9 a10 a11 b4 b5 b6 b7 b8 b9 b10 b11 a0 a1 a2 a3 b0 b1 b2 b3 c4 c5 c6 c7 c8 c9 c10 c11
P4[11:4] P3[3:0] P4[3:0] P5[11:4] P6[11:4]
d4 d5 d6 d7 d8 d9 d10 d11 c0 c1 c2 c3 d0 d1 d2 d3 e4 e5 e6 e7 e8 e9 e10 e11 f4 f5 f6 f7 f8 f9 f10 f11
P5[3:0] P6[3:0] P7[11:4] P8[11:4] P7[3:0] P8[3:0]
e0 e1 e2 e3 f0 f1 f2 f3 g4 g5 g6 g7 g8 g9 g10 g11 h4 h5 h6 h7 h8 h9 h10 h11 g0 g1 g2 g3 h0 h1 h2 h3
Buffer address
Data in receiver's buffer
MSB
LSB
P3[11:4] P2[3:0] P1[3:0] P2[11:4] P1[11:4]
00h
c11 c10 c9 c8 c7 c6 c5 c4 b3 b2 b1 b0 a3 a2 a1 a0 b11 b10 b9 b8 b7 b6 b5 b4 a11 a10 a9 a8 a7 a6 a5 a4
P6[11:4] P5[11:4] P4[3:0] P3[3:0] P4[11:4]
01h
f11 f10 f9 f8 f7 f6 f5 f4 e11 e10 e9 e8 e7 e6 e5 e4 d3 d2 d1 d0 c3 c2 c1 c0 d11 d10 d9 d8 d7 d6 d5 d4
P8[3:0] P7[3:0] P8[11:4] P7[11:4] P6[3:0] P5[3:0]
02h
h3 h2 h1 h0 g3 g2 g1 g0 h11 h10 h9 h8 h7 h6 h5 h4 g11 g10 g9 g8 g7 g6 g5 g4 f3 f2 f1 f0 e3 e2 e1 e0

Figure 78. The RAW12 data format reception

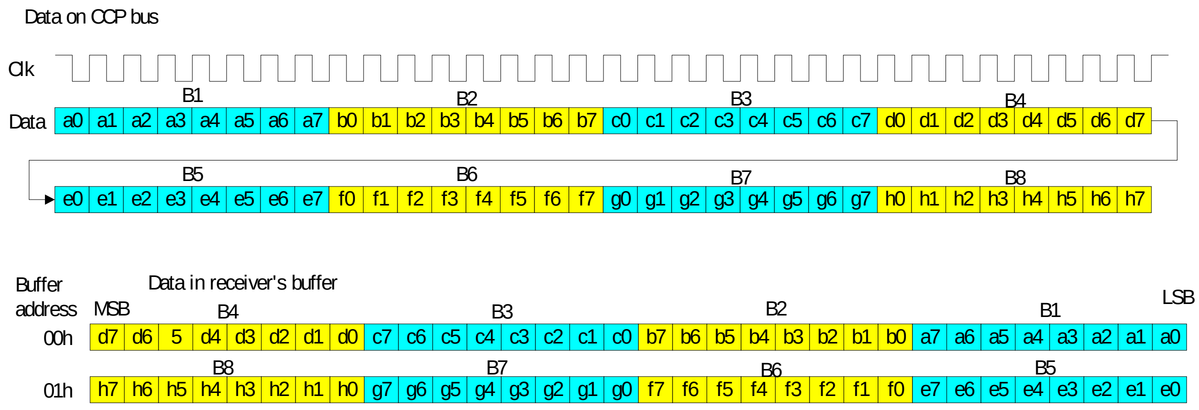

Data on CCP bus
Clk
Data
B1 B2 B3 B4
a0 a1 a2 a3 a4 a5 a6 a7 b0 b1 b2 b3 b4 b5 b6 b7 c0 c1 c2 c3 c4 c5 c6 c7 d0 d1 d2 d3 d4 d5 d6 d7
B5 B6 B7 B8
e0 e1 e2 e3 e4 e5 e6 e7 f0 f1 f2 f3 f4 f5 f6 f7 g0 g1 g2 g3 g4 g5 g6 g7 h0 h1 h2 h3 h4 h5 h6 h7
Buffer address
Data in receiver's buffer
MSB
LSB
B4 B3 B2 B1
00h
d7 d6 5 d4 d3 d2 d1 d0 c7 c6 c5 c4 c3 c2 c1 c0 b7 b6 b5 b4 b3 b2 b1 b0 a7 a6 a5 a4 a3 a2 a1 a0
B8 B7 B6 B5
01h
h7 h6 h5 h4 h3 h2 h1 h0 g7 g6 g5 g4 g3 g2 g1 g0 f7 f6 f5 f4 f3 f2 f1 f0 e7 e6 e5 e4 e3 e2 e1 e0

Figure 79. The compressed data format reception

9. Physical layer (SubLVDS)

The physical layer of CCP2 link is called SubLVDS. SubLVDS provides 1.8 V or 1.5 V-operated differential signaling for short distances. Modified LVDS type current mode transmitters / receivers should be used, and they should be optimized for maximum driving symmetry.
The CCP2 compliant SubLVDS supports maximum transfer speed of 416 MHz.

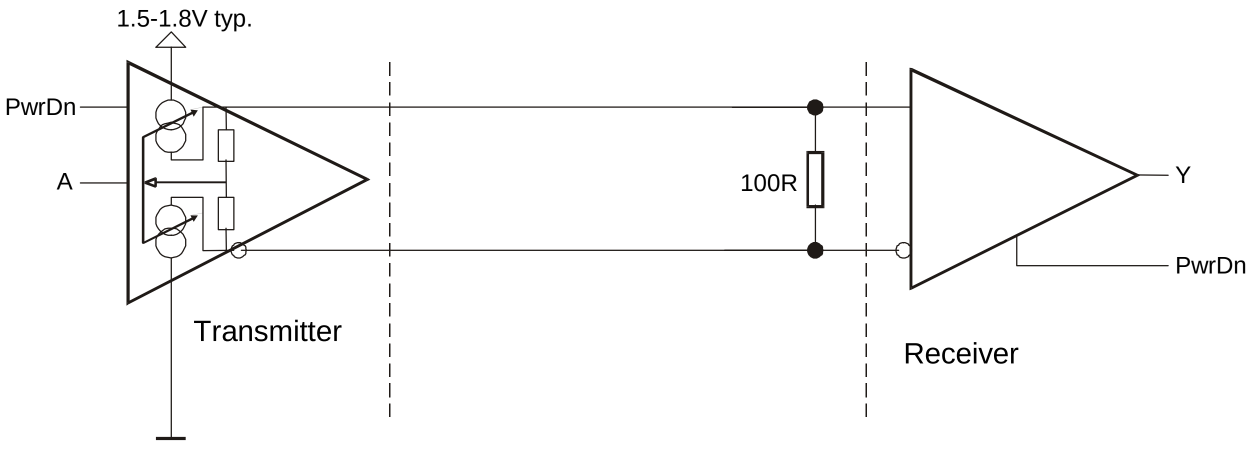


Figure 80. SubLVDS differential signaling, simplified diagram.

	Min	Typical	Max	Unit
Core operating voltage **V_{Core}**	1.0		1.95	V
Bus operating voltage **V_{DD}**	1.4	1.8	1.95	V
Fixed common mode voltage **V_{CMF}**	0.8	0.9	1.0	V

Table 13. DC characteristics

	Min	Max
Input	-0.5 V	**V_{DD}** + 0.4 V
Output	-0.5 V	**V_{DD}** + 0.5 V
ESD protection	2 kV	

Table 14. Absolute maximum ratings. Note: Values respect to GND.

	Rating	Unit
Storage temperature range	-65 to 150	°C
Junction temperature range	-40 to 125	°C
Ambient temperature range	-40 to 85	°C

Table 15. Temperature specification

9.1 Power down mode

Because of the static power consumption of the bus it has to be possible to shut down both the transmitter and receiver. Thus the I/O-cells must be equipped with **PwrDn** (internal) signals.

10. SubLVDS transmitter specification

The SubLVDS trasnmitter is self-biasing. The bias current is generated on-chip inside the transmitter cell. Ideally the bandgap voltage reference should be used for current set resistors.

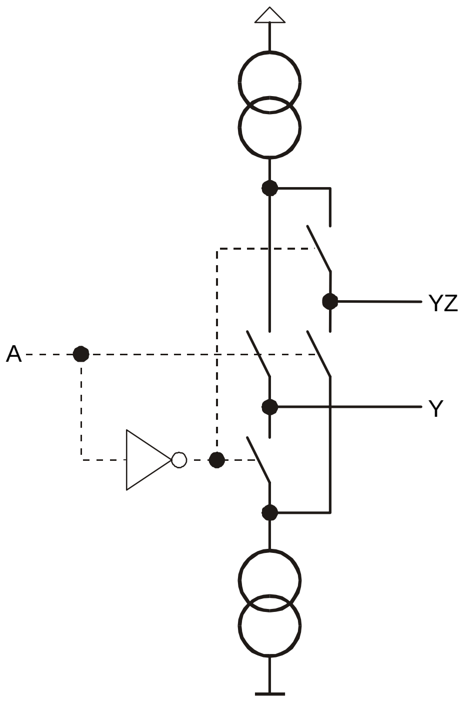


Figure 81. SubLVDS transmitter principle

Self biasing transmitter is able to adjust the line common mode voltage by detecting the $\mathbf{V_{CM}}$ and controlling the internal current source(s) accordingly (see Figure 80).

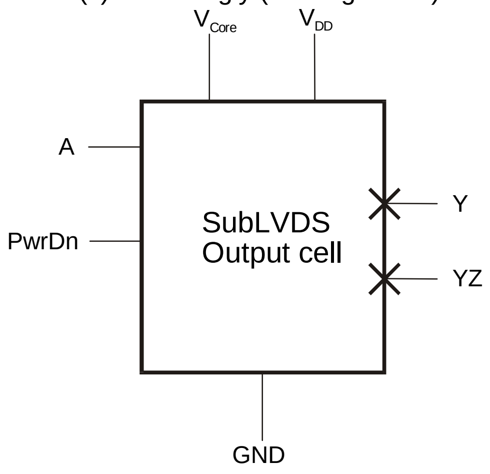


Figure 82. SubLVDS output cell connections (X = I/O pin, others internal)

In systems where there is a separate voltage for the IC core functions the SubLVDS output cell is powered by both the core operating voltage and the I/O operating voltage, core operating voltage being used for the internal signals **A** and **PwrDn**. In all other cases $\mathbf{V_{core}}$ = $\mathbf{V_{DD}}$.
The transmitted signal is understood as '1' when the current flows from the positive (non-inverting) terminal to the negative (inverting) terminal, and as '0' when the current flows in inverse direction.

Table 16. SubLVDS signals, transmitting end

Parameter	Signal levels			Unit
	Min	**Typical**	**Max**	
Fixed common mode voltage $\mathbf{V_{CMF}}$	0.8	0.9	1.0	V
Differential voltage swing $\mathbf{V_{OD}}$	100	150	200	mV
Drive current range	0.833	1.5	2	mA
Drive current variation $\mathbf{D_{IO}}$			15	%
Output impedance $\mathbf{R_{O}}$	40		140	Ω
Output impedance mismatch $\mathbf{D_{RO}}$			10	%
Clock duty cycle @ 416 MHz	45	50	55	%
$\mathbf{V_{OD}}$ rise time 20% - 80%	300		400	ps
$\mathbf{V_{OD}}$ fall time 80% - 20%	300		400	ps
Operating frequency	1		416	MHz
A/**PwrDn** low level input voltage	0		0.2	V
A/**PwrDn** high level input voltage	$\mathbf{V_{Core}}$ – 0.2		$\mathbf{V_{Core}}$	V
Power up / down time (**PwrDn**)			20	µs
Power supply rejection ratio (**PSRR**) 0-100 MHz	30			dB
Power supply rejection ratio (**PSRR**) 100-1000 MHz	10			dB

Definitions:
$\mathbf{V_{CMF}}$: Fixed common mode voltage of the transmission line transmitter end
$\mathbf{V_{OD}}$: Differential voltage swing measured over 100 Ω resistance connected directly between the transmitter differential pins
Drive current range: Minimum and maximum values between which the drive current can be set using an external or on-chip resistor.
$\mathbf{D_{IO}}$: Drive current variation from ideal value over temperature with a constant external resistor value
$\mathbf{R_{O}}$: Driver output impedance (single line)
$\mathbf{D_{RO}}$: Mismatch of the output impedance between the lines of a single differential pair
PSRR : Nominal value for the inteference at $V_{\mathbf{CM}}$ voltage through digital supply relative to the interference at digital supply over the 0-1 GHz operating range.
PSRR = 20*log10 ($V_{\mathbf{DDinterferece}}$ (peak-to-peak) / $V_{\mathbf{Cminterference}}$ (peak-to-peak))

10.1 Transmitter cell current consumption specification

The current consumption in SubLVDS transmitter cell is mostly static, but there exists also frequency dependency. The DC and AC current consumption figures are presented in the following table.

Table 17. AC and DC current consumption

Parameter	Signal levels			Unit
	Min	**Typical**	**Max**	
DC current consumption	-	3	4	mA
Maximum peax current	-	-	8	mA
AC current consumption		2.5	3.5	µA/MHz

The total current consumption is solved from the equation below.

$$I_{tx}(total) = I_{tx}(DC) + I_{tx}(AC) * f$$

f = frequency of operation

11. Receiver specification

The SubLVDS receiver is self-biasing. The bias current is generated on-chip inside the receiver cell. Ideally the bandgap voltage reference should be used for current set resistors. The termination resistor is typically a separate component. However, termination can be done on-chip, if the accuracy of termination resistor is guaranteed to be according to specification.

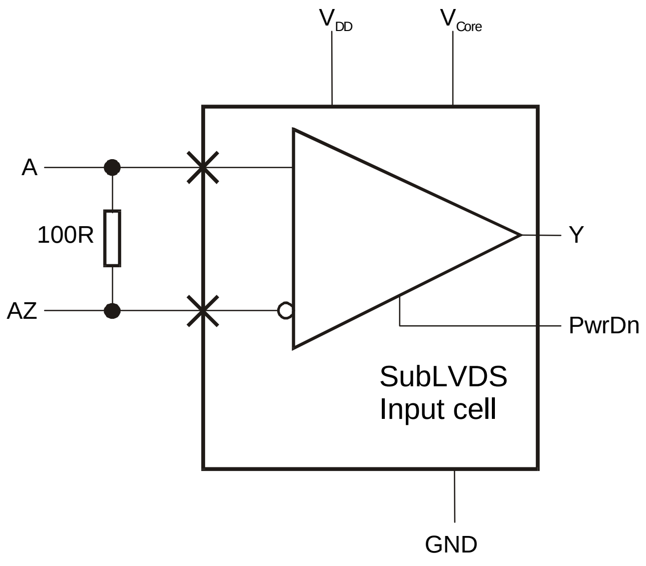


Figure 83. SubLVDS input cell connections (X = I/O pin, others internal)

In systems where there is a separate voltage for the IC core functions the SubLVDS output cell is powered by both the core operating voltage and the I/O operating voltage, core operating voltage being used for the internal signals **Y** and **PwrDn**. In all other cases $\mathbf{V_{core}}$ = $\mathbf{V_{DD}}$.
The received signal is understood as '1' when the voltage in positive (non-inverting) terminal is more than $\mathbf{V_{THH}}$ above $\mathbf{V_{CM}}$, and the negative (inverting) terminal voltage is more than $\mathbf{V_{THL}}$ below $\mathbf{V_{CM}}$. The signal is understood as '0' when the voltage between the terminals is vice versa.

Parameter	Signal levels			Unit
	Min	**Typical**	**Max**	
Input voltage range	($\mathbf{V_{CMF}}$)-0.4	$\mathbf{V_{CMF}}$ (0.9V)	($\mathbf{V_{CMF}}$)+0.4	V
Receiver input low threshold $\mathbf{V_{THL}}$	-25			mV
Receiver input high threshold $\mathbf{V_{THH}}$			25	mV
Termination resistor value	80	100	120	Ω
Operating frequency	1		416	MHz
Y/**PwrDn** low level output voltage	0		0.1	V
Y/**PwrDn** high level output voltage	$\mathbf{V_{Core}}$ – 0.1		$\mathbf{V_{Core}}$	V
Power up / down time (**PwrDn**)			20	µs

Table 18. SubLVDS signals, receiving end

Definitions:
Input voltage range: Minimum and maximum voltages in the **A** and **AZ** lines
Receiver input low & high threshold: These define the receiver sensitivity. When both low and high input thresholds are exceeded the output **Y** is defined.

11.1 Termination

Basic termination is a 100 Ω resistance between + and - signals in the receiving end. Termination resistance value may vary slightly for optimization purposes, depending on the case. See Table 18 for appropriate values.

11.2 Receiver cell current consumption specification

The current consumption in SubLVDS receiver cell is mostly static, but there exists also frequency dependency. The DC and AC current consumption figures are presented in the following table.

Table 19. SubLVDS Receiver cell AC and DC current consumption

Parameter	Signal levels			Unit
	Min	**Typical**	**Max**	
DC current consumption	-	1.5	2	mA
Maximum peak current	-	-	4	mA
AC current consumption		1.25	1.75	µA/MHz

The total current consumption is solved from the equation below.

$$I_{rx}(total) = I_{rx}(DC) + I_{rx}(AC) * f$$

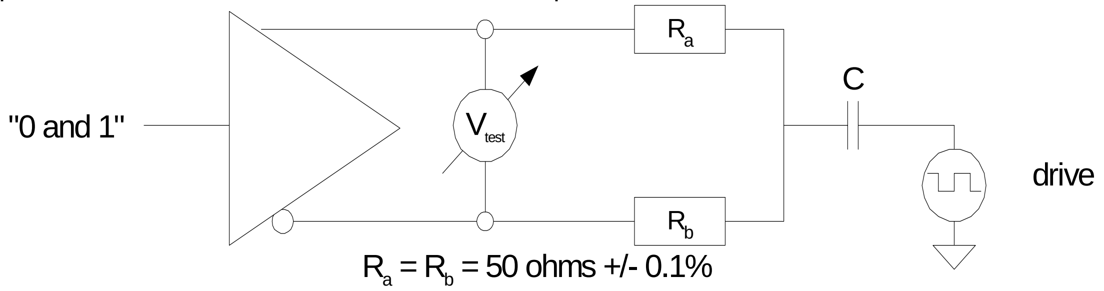
f = frequency of operation

12. Transmitter cell output impedance specification

The IEEE STD 1596.3-1996 specifies two different test methodologies for testing the transmitter cell differential output impedance, the dynamical output impedance and static output impedance.
The test methodologies are presented in the following chapters. The chapters are almost a direct copy of the IEEE standard. The circuit analysis is equivalent for both SubLVDS and LVDS. The results of impedance measurements are presented for both cases for clarity.

12.1 AC driver-output impedance

The Figure 84illustrates the means for testing the driver's dynamic output impedance. Dynamic output impedance is important in the very high frequency operation and is therefore essential quantity to measure. Because SubLVDS uses different signal levels than LVDS, the driving value and the acceptance criteria is modified. The differences are presented in the Table 20.



$R_a = R_b$ = 50 ohms +/- 0.1%
C = 0.033 uF +/- 20%
drive = 200 mV peak-to-peak, 10 MHz
$|V_{test}|$ = 20 mV peak-to-peak

Figure 84. Driver dynamic impedance output test.

Parameter	LVDS	SubLVDS
Drive	500 mV	200 mV
V_{test}	50 mV	20 mV

Table 20. LVDS signal levels vs. SubLVDS signal levels.

12.2 DC driver-output impedance

The Figure 85below illustrates a means of measuring the static driver output impedances, which have min/max as well as mismatch constraints. For each of the two possible output signal values (0 and 1), the output voltages shall be measured with driver-load common-mode (V_{cm}) voltages of V_{cmlo} andV_{cmhi}, yielding measured voltages of V_{oalo}, V_{oblo}, V_{oahi} and V_{obhi} .
The values of these output voltages are based on the absolute and relative impedances of the drivers. The difference in a single output voltage (for V_{cmlo} and V_{cmhi} output-load voltages) is affected by the value of the driver's output impedance on this signal. The difference in the differential voltages (for V_{cmlo} and V_{cmhi} output-load voltages) is affected by the matching of the driver's a and b output impedances.

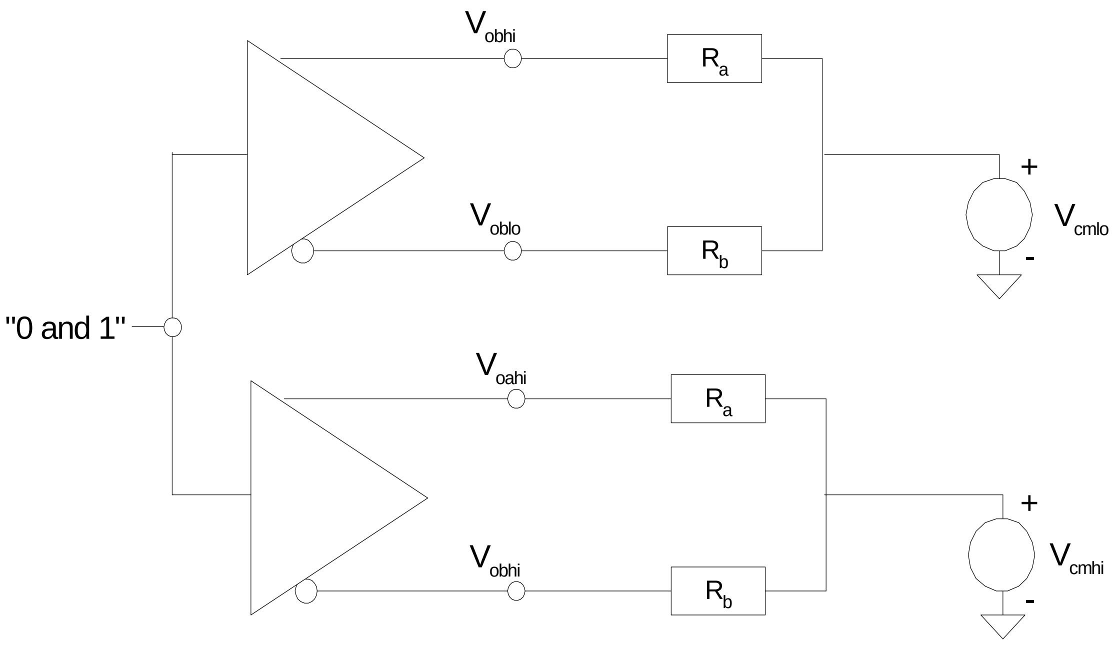


Figure 85. Driver static impedance output test.

Parameter	LVDS	SubLVDS
V_{cmlo}	1.0V	0.8 V ($V_{\mathbf{CMF}}$ – 0.1V)
V_{cm}	1.2V	0.9 V ($V_{\mathbf{CMF}}$)
V_{cmhi}	1.4V	1.0 V ($V_{\mathbf{CMF}}$ + 0.1V)

Table 21. V_{cmlo}, V_{cm} and V_{cmhi} values

12.3 Circuit analysis

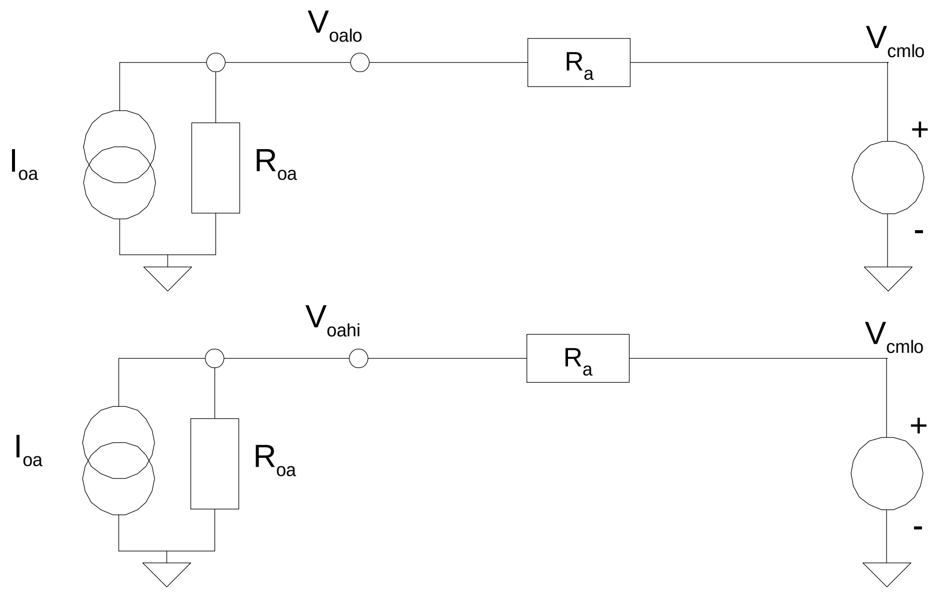


Figure 86. Equivalent circuits for V_{oalo} and V_{oahi}

The equations for the above 2 equivalent circuits are

$$I_{oa} = \frac{V_{oalo}}{R_{oa}} + \frac{V_{oalo} - V_{cmlo}}{R_a}$$
$$I_{oa} = \frac{V_{oalo}}{R_{oa}} + \frac{V_{oalo} - V_{cmlo}}{R_a}$$

Equating the above 2 equations gives

$$\frac{V_{oalo}}{R_{oa}} + \frac{V_{oalo}}{R_a} - \frac{V_{cmlo}}{R_a} = \frac{V_{oahi}}{R_{oa}} + \frac{V_{oahi}}{R_a} - \frac{V_{cmhi}}{R_a}$$

Re--organising gives

$$\left(V_{oahi} - V_{oalo}\right)\left(\frac{1}{R_{oa}} + \frac{1}{R_a}\right) = \frac{V_{cmhi} - V_{cmlo}}{R_a}$$

Solving for (V_{oahi} - V_{oalo}) yields

$$\left(V_{oahi} - V_{oalo}\right) = \frac{V_{cmhi} - V_{cmlo}}{\left(\frac{Ra}{R_{oa}} + 1\right)}$$

Solving for R_{oa} yields

$$R_{oa} = \frac{R_a}{\left(\frac{V_{cmhi} - V_{cmlo}}{V_{oahi} - V_{oalo}} + 1\right)}$$

The equations for (V_{obhi} - V_{oblo}) and R_{ob} are

$$\left(V_{obhi} - V_{oblo}\right) = \frac{V_{cmhi} - V_{cmlo}}{\left(\frac{R_b}{R_{ob}} + 1\right)}$$

$$R_{ob} = \frac{Rb}{\left(\frac{V_{cmhi} - V_{cmlo}}{V_{obhi} - V_{oblo}} + 1\right)}$$

The following tables illustrate the relationship between R_{oa} and (V_{oahi} - V_{oalo}) for R_{oa} in the range 40 ohms to 140 ohms for both the LVDS and SubLVDS cases.

R_{oa} (ohms)	LVDS (V_{oahi}-V_{oalo}) (mV)	SubLVDS (V_{oahi}-V_{oalo}) (mV)
40	178	89
50	200	100
60	218	109
70	233	117
80	246	123
90	257	129
100	267	133
110	275	138
120	282	141
130	289	144
140	295	147

Table 22. R_{oa} Versus (V_{oahi}-V_{oalo})

SubLVDS (V_{oahi}-V_{oalo}) (mV)	R_{oa} (ohms)
80	33.3
90	40.9
100	50.0
110	61.1
120	75.0
130	92.9
140	116.7
150	150.0

Table 23. SubLVDS (V_{oahi}-V_{oalo}) Versus R_{oa}

LVDS (V_{oahi}-V_{oalo}) (mV)	R_{oa} (ohms)
160	33.3
170	37.0
180	40.9
190	45.2
200	50.0
210	55.3
220	61.1
230	67.6
240	75.0
250	83.3
260	92.9
270	103.8
280	116.7
290	131.8
300	150.0

Table 24. LVDS (V_{oahi}-V_{oalo}) Versus R_{oa}

12.4 LVDS and SubLVDS Output impedance test specifications

Parameter	Correspnds to	Units	Minimum	Maximum
V_{oahi} - V_{oalo}	R_O terminal a	mV	178 (based on R_{oa} = 40 ohms)	295 (based on R_{oa} = 140 ohms)
V_{obhi} - V_{oblo}	R_O terminal b			
$\|(V_{oah} - V_{obhi}) – (V_{oalo} - V_{oblo})\|$	10% matching of reflection coefficients	mV	0	20

Table 25. LVDS Static Load output Voltage Constraints

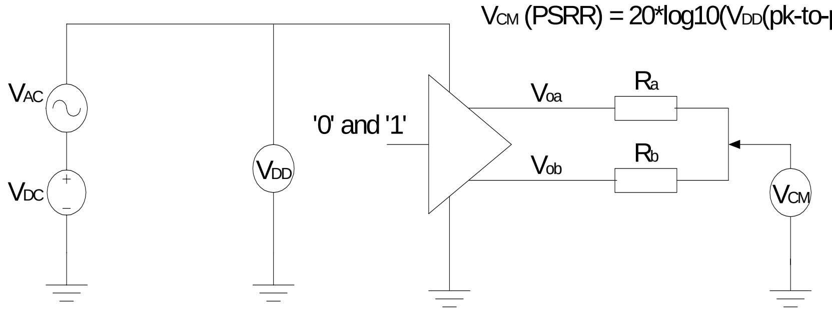
Parameter	Correspnds to	Units	Minimum	Maximum
V_{oahi} - V_{oalo}	R_O terminal a	mV	89 (based on R_{oa} = 40 ohms)	147 (based on R_{oa} = 140 ohms)
V_{obhi} - V_{oblo}	R_O terminal b			
$\|(V_{oah} - V_{obhi}) – (V_{oalo} - V_{oblo})\|$	10% matching of reflection coefficients	mV	0	20

Table 26. SubLVDS Static Load output Voltage Constraints

12.5 Power supply rejection ratio test specification

The PSRR test method is illustrated in the Figure 87 below. The test system consists of DC power supply, signal generator and appropriate circuitry to prevent the interference signal propagation to the power supply. If DC offset voltage is available in the signal generator, it is also possible to attach the signal generator directly to the SubLVDS cell's operating voltage pin. The nominal values for test circuitry components (except for the termination resistor) depend on the operating frequency. The PSRR has to be measured over 0-1Ghz operating range.

V_{CM} (PSRR) = 20*log10(V_{DD}(pk-to-pk)/V_{CM}(pk-to-pk))



R_a = R_b = 50 ohms +/- 0.1%
V_{DD} = 100 mV peak-to-peak sine wave centered around 1.8V DC
V_{CM} = less than 3.16 mV peak-to-peak sine wave

Figure 87. PSRR test method

13. PWB Design considerations

The design principles for implementing SubLVDS transmitter/receiver on PWB conform to LVDS design principles. A short summary of general principles is presented here; more complete information can be found e.g. from [**3**].

In a multilayer board, use dedicated V_{CC} and ground planes and place LVDS signals to a different layer than TTL signals. If this is not possible (e.g. in one layer flex connection) the following kind of arrangement should be used:

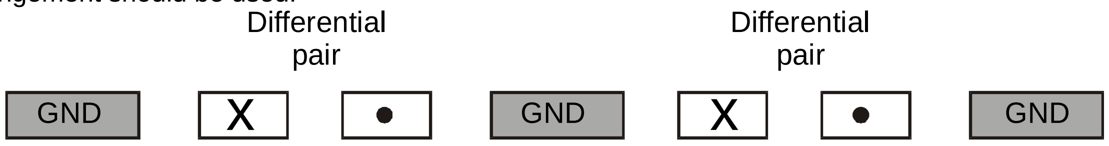


Figure 88. SubLVDS link wiring principle in 1-layer parts

The traces for differential pair in PWB should have equal lengths. The skew between traces creates phase difference between voltages and is radiated as common-mode noise.

The traces for data and clock signals should also have equal lengths to ensure reliable timing.

Transmission line impedance should match the used SubLVDS termination impedance, which nominally is 100 Ω but can be varied

14. References

1. JPEG & JFIF http://www.jpeg.org/
2. EXIF http://www.exif.org/
3. LVDS Owner's Manual. National Semiconductor, http://www.national.com/appinfo/lvds/0,1798,100,00.html
4. I2C standard (v2.1 January 2000). Philips Semiconductor

ANNEX