Developer's Guide

Timing Controller Solutions – Generation 2 for Pervasive Displays 4.41″, 7.4" and 10.2″ Panels

TCM2-P441-231_v1.1, TC2-P441-231_v1.1, TCM2-P74-231_v1.0, TC2-P74-231_v1.0, TCM2-P102-231_v1.2

Classification: **Public**

Document Revision: **E**

Table of Contents

1 Introduction

ePaper **Timing Controller Solutions Generation 2** (TCS2) provide timing controller functionality for **Pervasive Displays**' large size panels (**4.41″, 7.4" and 10.2″**). Solution for each of the panels provides identical core functionality, command set and physical interface. Offered as a chip only (**Timing Controller – TC**) or as fully-assembled PCB module (**Timing Controller Module – TCM**), the solution allows a quick and easy integration with your host system, minimizing the cost and time-to-market.

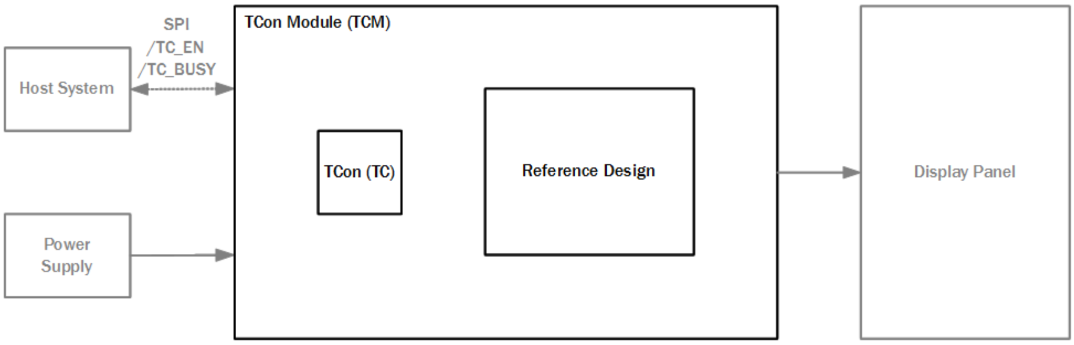


Figure 1.1: TCS block diagram

TCS2 products can be connected to a host system via fast and reliable Serial Peripheral Interface (SPI). TCS2 is controlling both the source and gate drivers, composing waveforms required to generate high quality images on the display.

1.1 Ordering Information

Product Family:	Timing Controller Solutions Generation 2 (TCS2)
Product Line:	TCS2 for Pervasive Displays (TCS2-P)

TCS2-P441-231 Part Numbers

Timing Controller Module:	**TCM2-P441-231_v1.1**
Timing Controller:	**TC2-P441-231_v1.1**

TCS2-P74-231 Part Numbers

Timing Controller Module:	**TCM2-P74-231_v1.0**
Timing Controller:	**TC2-P74-231_v1.0**

TCS2-P102-231 Part Numbers

Timing Controller Module:	**TCM2-P102-231_v1.2**
Timing Controller:	***Not Available as TCon***

1.2 Compatibility Note

Timing Controller Solutions Generation 2 products supersede the respective Generation 1 products, being fully backwards compatible[1] in terms of host-side hardware and software.

The TC reference designs have changed in Generation 2 and need to be adopted accordingly – this applies only to the users utilizing TCs integrated into their own designs.

1.2.1 Superseded Products Part Numbers

Product Family:	Timing Controller Solutions (TCS)
Product Line:	TCS for Pervasive Displays (TCS-P)

TCS-P441 & TCS2-P441 Part Numbers

Timing Controller Module:	TCM-P441-110_v1.1; TCM2-P441-231_v1.0
Timing Controllers:	TC-P441-110_v1.1; TC2-P441-231_v1.0

TCS-P74 Part Numbers

Timing Controller Module:	TCM-P74-110_v1.1; TCM-P74-220_v1.1
Timing Controllers:	TC-P74-110_v1.1; TC-P74-220_v1.1

TCS2-P102 Part Numbers

Timing Controller Modules:	TCM2-P102-231_v1.1

1.3 Supported Display Panels

Each TCS version is compatible with a given PDI ePaper display part number listed in the Table 1.1 below and is not compatible with any other part number.

Please consult the respective PDI display *Product Specification* document for information on the display usage and precautions.

NOTE Please mind the limitations related to the displays being susceptible to direct sunlight and strong artificial light.

MpicoSys TCS Product Code	PDI Display Size	PDI Display Material (FPL)	PDI Display Part No.	PDI Display Resolution [px]	PDI Display Density [dpi]
TCS2-P441-231_v1.1	4.41″	Aurora Mb – v231	E1441CS021	400×300	113
TCS2-P74-231_v1.0	7.4″		EW074CT011	480×800	126
TCS2-P102-231_v1.2	10.2″		EZ102CT011	1024×1280	160

Table 1.1: Supported display panels summary

1.4 Features

- Supporting the state-of-the-art PDI Aurora Mb panels
- SPI interface to host – slave device with additional /TC_EN and /TC_BUSY lines
- 1-bit color (black and white)

[1] Exception: GetSensorData command – see section 5.7.5.1

- Complete solution including:
  - Temperature compensation
  - Common electrode voltage compensation
  - All voltages needed for the display
- Internal image buffer retains content during system power down
- Backward-compatible with Generation 1

New In Generation 2

- Direct display update – no flashing during the image transition
- Partial image upload – no need to send the full image data
- Reduced power consumption due to reactive implementation
- Multiple image slots
- Image data checksum calculation ensuring data integrity
- Display update temperature override and block driving control[1)]
- Maximum slot number increased to 99 (custom build with 128 Mbit memory)[1)]

1.4.1 Block Driving

Block Driving is a TCS2-P102-231 display driving method applied when the temperature measured by the TCS2 is higher than 27.5 °C. Block Driving reduces the crosstalk effect, but increases the display refresh time and increases the display panel susceptibility to direct sunlight and strong artificial light: display panel exposed to intense sunlight has lower contrast ratio due to photosensitivity of the pixel transistors. To counteract the sunlight effect, block driving is disabled by default in TCS2-P102-231_v1.2.

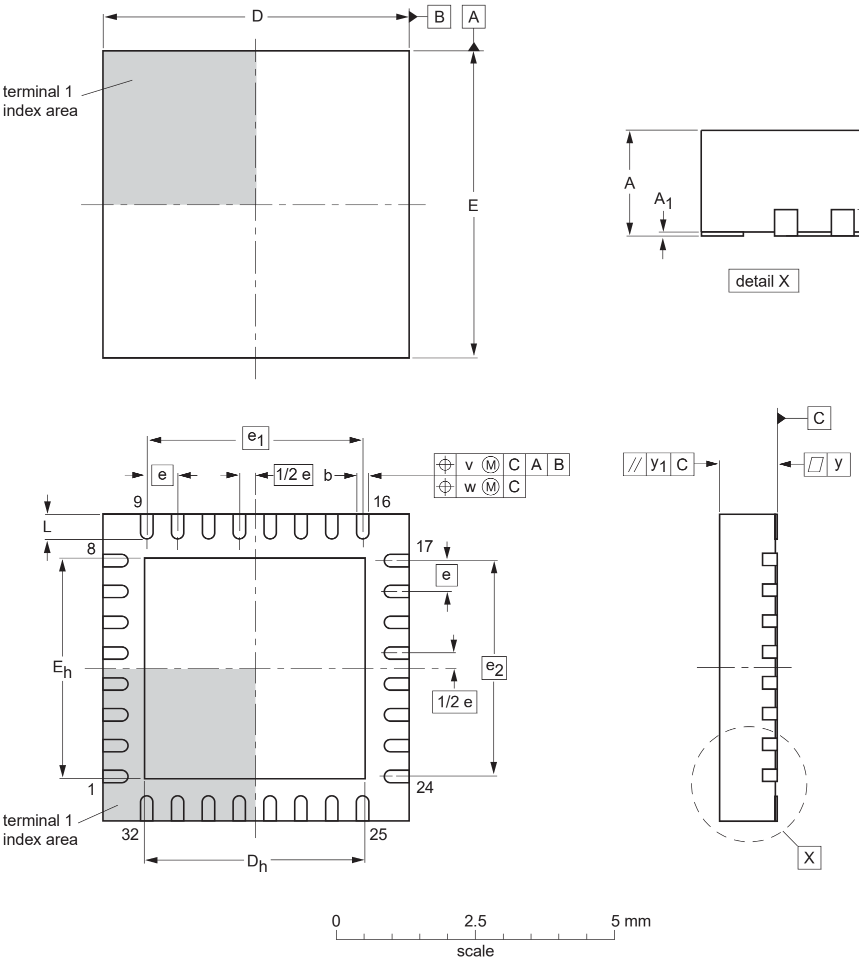
1) Only in TCS2-P102-231_v1.2.

2 Outline

2.1 TC2

The information below applies to TC2-P441-231 and TC2-P74-231.[1]

HVQFN33: plastic thermal enhanced very thin quad flat package; no leads; 32 terminals; body 5 x 5 x 0.85 mm



Unit		A[2]	A_1	b	c	D[2]	D_h	E[2]	E_h	e	e_1	e_2	L	v	w	y	y_1
mm	max		0.05	0.30		5.1	3.75	5.1	3.75				0.5				
	nom	0.85			0.2					0.5	3.5	3.5		0.1	0.05	0.05	0.1
	min		0.00	0.18		4.9	3.45	4.9	3.45				0.3				

Table 2.1: Dimensions (mm are the original dimensions)

[1] © NXP Semiconductors N.V. 2014. All rights reserved.
[2] Plastic or metal protrusions of 0.075 mm maximum per side are not included.

2.2 TCM2

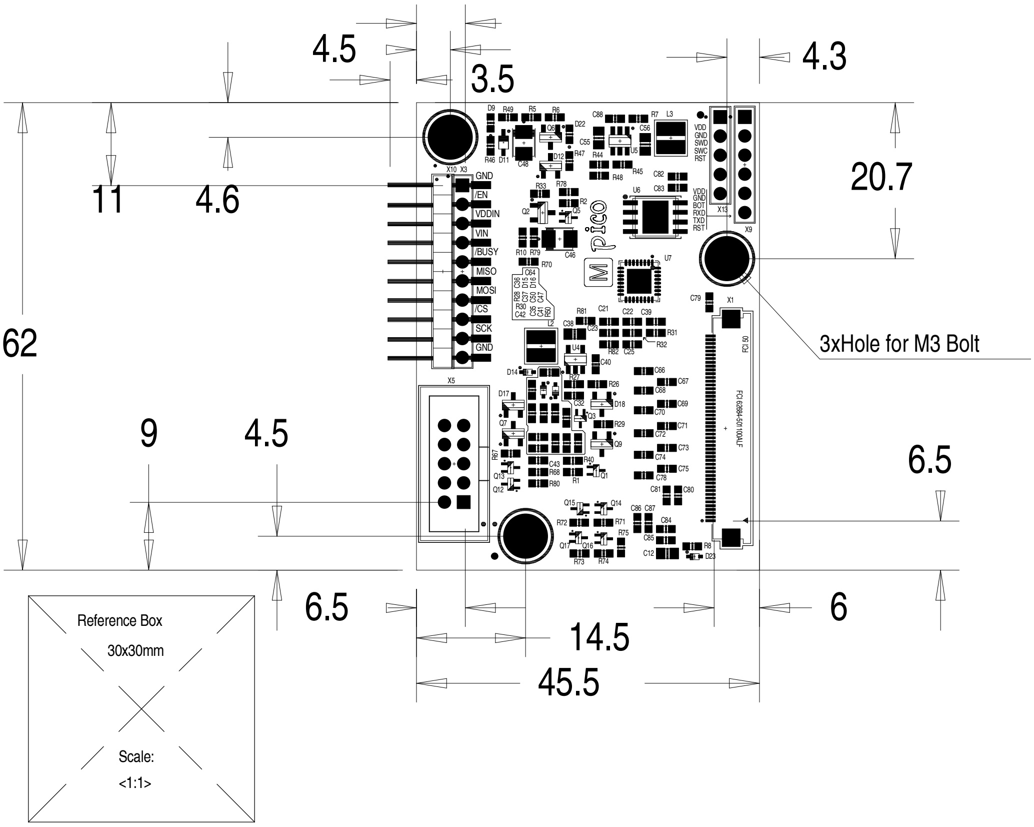


Figure 2.1: TCM2-P441-231 Outline (all dimensions in mm)

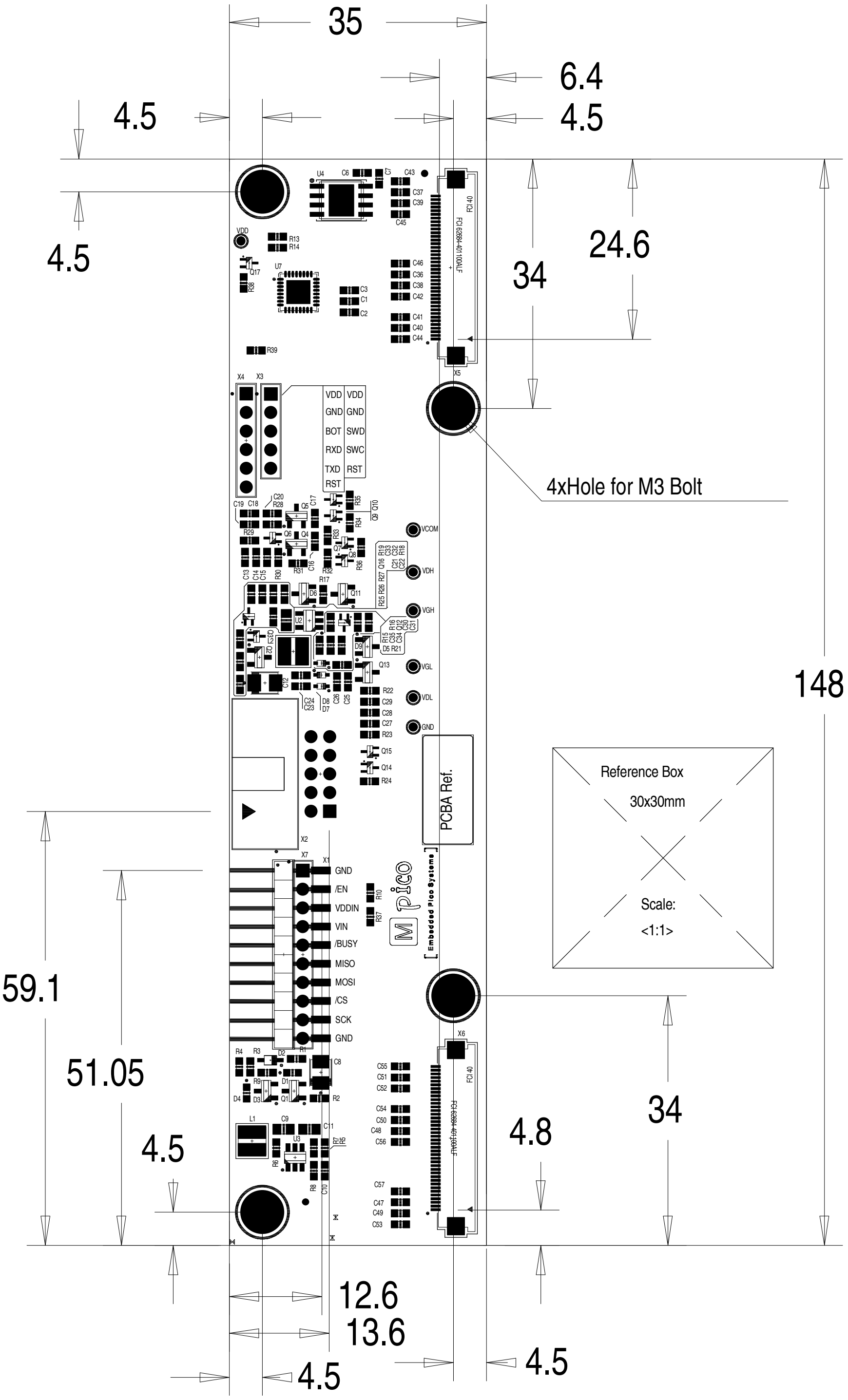


Figure 2.2: TCM2-P74-231 Outline (all dimensions in mm)

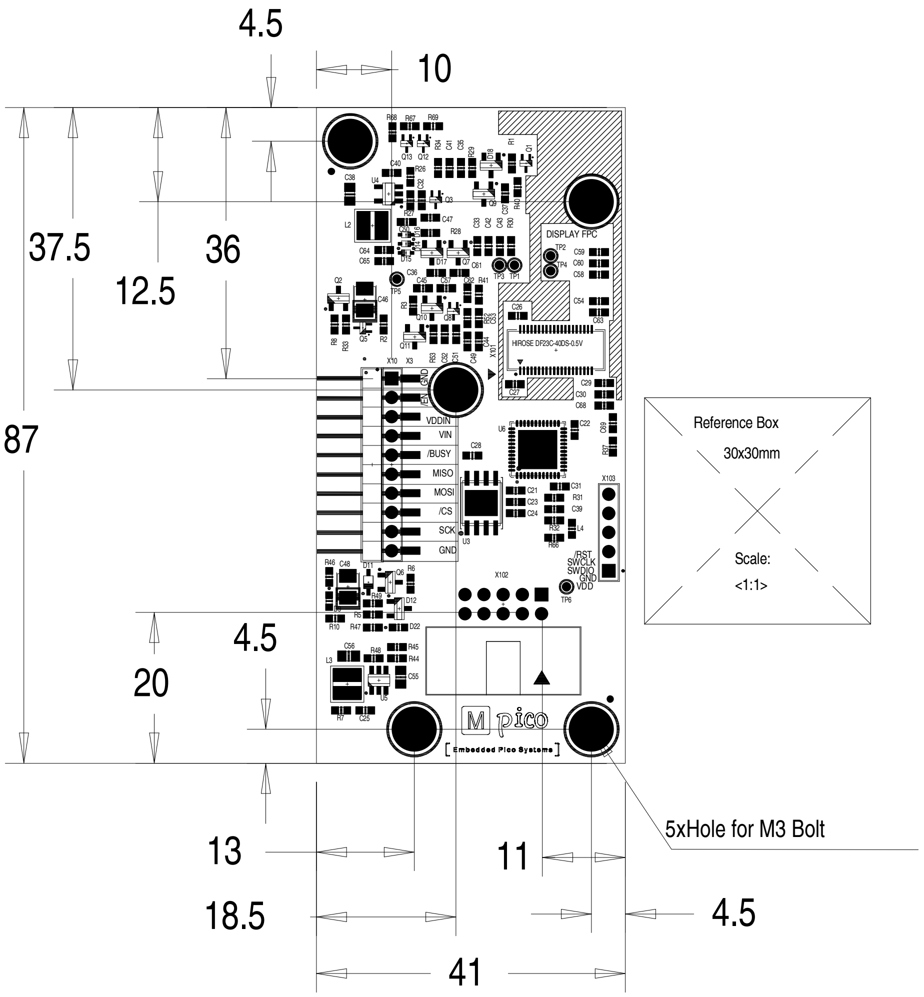


Figure 2.3: TCM2-P102-231 Outline (all dimensions in mm)

3 Characteristics

Unless specified otherwise, the values in this chapter are applicable to the whole TCS2 product family, i.e. to both TC2 and TCM2.

3.1 Operating Conditions

TCS2-P441-231

Symbol	Description	Min	Typ	Max	Unit
VDDIN / VIN[1)]	Standard operating voltage[1)]	2.7	3.0	3.3	V
T_{op}	Operating temperature	0	+21	+50	°C

Table 3.1: Typical operating conditions – TCS2-P441-231

TCS2-P74-231

Symbol	Description	Min	Typ	Max	Unit
VDDIN	Standard digital operating voltage	2.7	3.0	3.3	V
VIN	Standard analog operating voltage	2.7	3.0	3.3	V
T_{op}	Operating temperature	0	+21	+50	°C

Table 3.2: Typical operating conditions – TCS2-P74-231

TCS2-P102-231

Symbol	Description	Min	Typ	Max	Unit
VDDIN	Standard digital operating voltage	2.7	3.0	3.6	V
VIN	Standard analog operating voltage	2.0	3.0	5.5	V
T_{op}	Operating temperature	0	+21	+50	°C

Table 3.3: Typical operating conditions – TCS2-P102-231

3.2 Absolute Maximum Ratings

NOTE TCM2 features solder pads for overvoltage protection 3.6 V Zener diode (D11 in TCM2-P441 and TCM2-P102, D2 in TCM2-P74.) The diode is by default not mounted to limit the TCM2 current consumption. If required, the diode can be mounted in the designated spot at the User's own account. It is recommended to use BZX384 3V6 diode. This will increase the average current consumption by 1 mA during all operations.

TCS2-P441-231

Symbol	Description	Min	Typ	Max	Unit
VDDIN / VIN[1)]	Supply voltage[1)]	−0.5	–	4.0	V
V_I[2)]	Logic input voltage[2)]	−0.5	–	4.6	V
T_{st}	Storage temperature	−20	–	+60	°C

Table 3.4: Absolute maximum ratings – TCS2-P441-231

1) Due to the fact TCS2-P441-231 utilizes internal display panel charge pump, the supply pins VDDIN and VIN are shorted with a 0R resistor.
2) Only valid when the VDDIN supply voltage is present.

TCS2-P74-231

Symbol	Description	Min	Typ	Max	Unit
VDDIN	Digital supply voltage	−0.5	–	4.0	V
VIN	Analog supply voltage	−0.5	–		V
V_I[2)]	Logic input voltage[2)]	−0.5	–	4.6	V
T_{st}	Storage temperature	−20	–	+60	°C

Table 3.5: Absolute maximum ratings – TCS2-P74-231

TCS2-P102-231

Symbol	Description	Min	Typ	Max	Unit
VDDIN	Digital supply voltage	−0.5	–	4.0	V
VIN	Analog supply voltage	−0.5	–	6.0	V
V_I[2)]	Logic input voltage[2)]	−0.3	–	VDDIN + 0.3	V
T_{st}	Storage temperature	−20	–	+60	°C

Table 3.6: Absolute maximum ratings – TCS2-P102-231

3.3 Endurance

TCS2 products are limited to 100,000 display update use cycles per frame buffer slot resulting from flash memory read/write cycles limitation.

3.4 TCM2 Supply Current Characteristics

Measurement Setup

Current consumption measured with Agilent 34411A Multimeter;

VDDIN shorted with VIN; range from 2.7 V to 3.3 V.

NOTE Values vary with ambient temperature, supply voltage, the displayed pattern, and the host controller settings.

TCS2-P441-231

Symbol	Description	Operation	Min	Max	Unit
IDD	Average current consumption	Display update – Quality	5.7	7.3	mA
		Display update – Flashless	6.9	8.7	mA
		Display update – Flashless-Inverted	7.3	12.8	mA
		Data reception on SPI	5.5	6.5	mA
		Disabled (/TN_EN inactive)	<1	<1	µA
E	Average energy consumption in room temperature	Display update – Quality	56	72	mJ
		Display update – Flashless	23.5	36	mJ
		Display update – Flashless-Inverted	45	79	mJ

Table 3.7: Supply current characteristics – TCS2-P441-231

TCS2-P74-231

Symbol	Description	Operation	Min	Max	Unit
IDD	Average current consumption	Display update – Quality	21.6	35.4	mA
		Display update – Flashless	15.5	32.4	mA
		Display update – Flashless-Inverted	24.8	60.4	mA
		Data reception on SPI	4.4	4.9	mA
		Disabled (/TN_EN inactive)	<1	<1	µA
E	Average energy consumption in room temperature	Display update – Quality	228	332	mJ
		Display update – Flashless	78	82	mJ
		Display update – Flashless-Inverted	197	207	mJ

Table 3.8: Supply current characteristics – TCS2-P74-231

TCS2-P102-231

Symbol	Description	Operation	Min	Max	Unit
IDD	Average current consumption	Display update – Quality	46.5	173.2	mA
		Display update – Flashless	37.3	480.9	mA
		Display update – Flashless-Inverted	58.6	510.4	mA
		Data reception on SPI	7.8	8.2	mA
		Disabled (/TN_EN inactive)	<1	<1	µA
E	Average energy consumption in room temperature	Display update – Quality	410	1194	mJ
		Display update – Flashless	150	1577	mJ
		Display update – Flashless-Inverted	389	2839	mJ

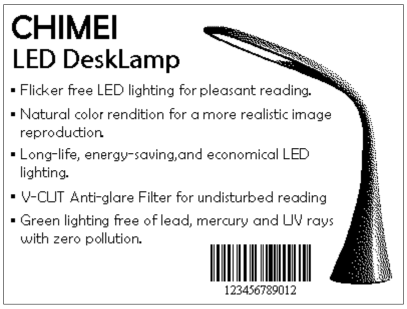
Table 3.9: Supply current characteristics – TCS2-P102-231

Measurement Results Conditions

The below table describes conditions at which the results from tables above were achieved. *ESL* images are presented below the table. *Checkerboard* image is a 1 pixel by 1 pixel black and white checkerboard fulfilling the whole display area.

Measurement		Value	Power Supply (VDD = VIN) [V]	Image Used for Measurement	Ambient Temp. [°C]
Average current consumption	Display update	Min	3.3	Transition *ESL* to *ESL*	21
		Max	2.7	Transition *Checkerboard* to *Checkerboard*	21
	Data reception on SPI	Min	2.7	*ESL*	21
		Max	3.3	*Checkerboard*	21
Average energy consumption in room temperature	Display update	Min	2.7	ESL	21
		Max	3.3	*Checkerboard*	21

Table 3.10: Measurement results conditions

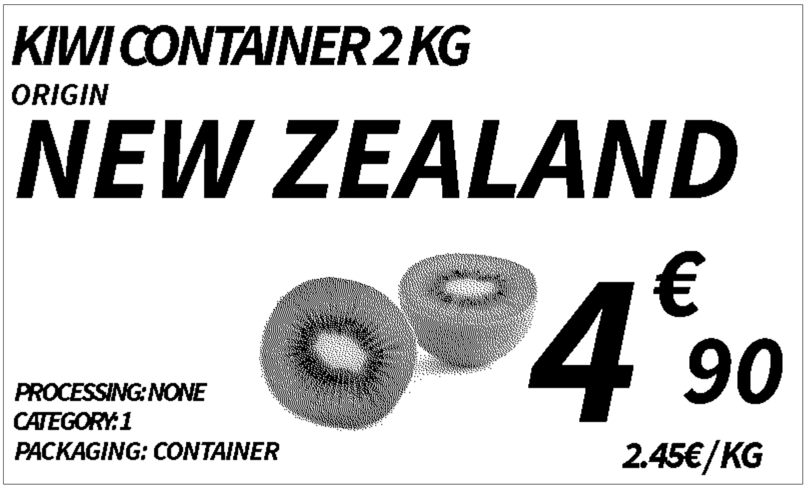


CHIMEI
LED DeskLamp

- Flicker free LED lighting for pleasant reading.
- Natural color rendition for a more realistic image reproduction.
- Long-life, energy-saving,and economical LED lighting.
- V-CUT Anti-glare Filter for undisturbed reading
- Green lighting free of lead, mercury and UV rays with zero pollution.

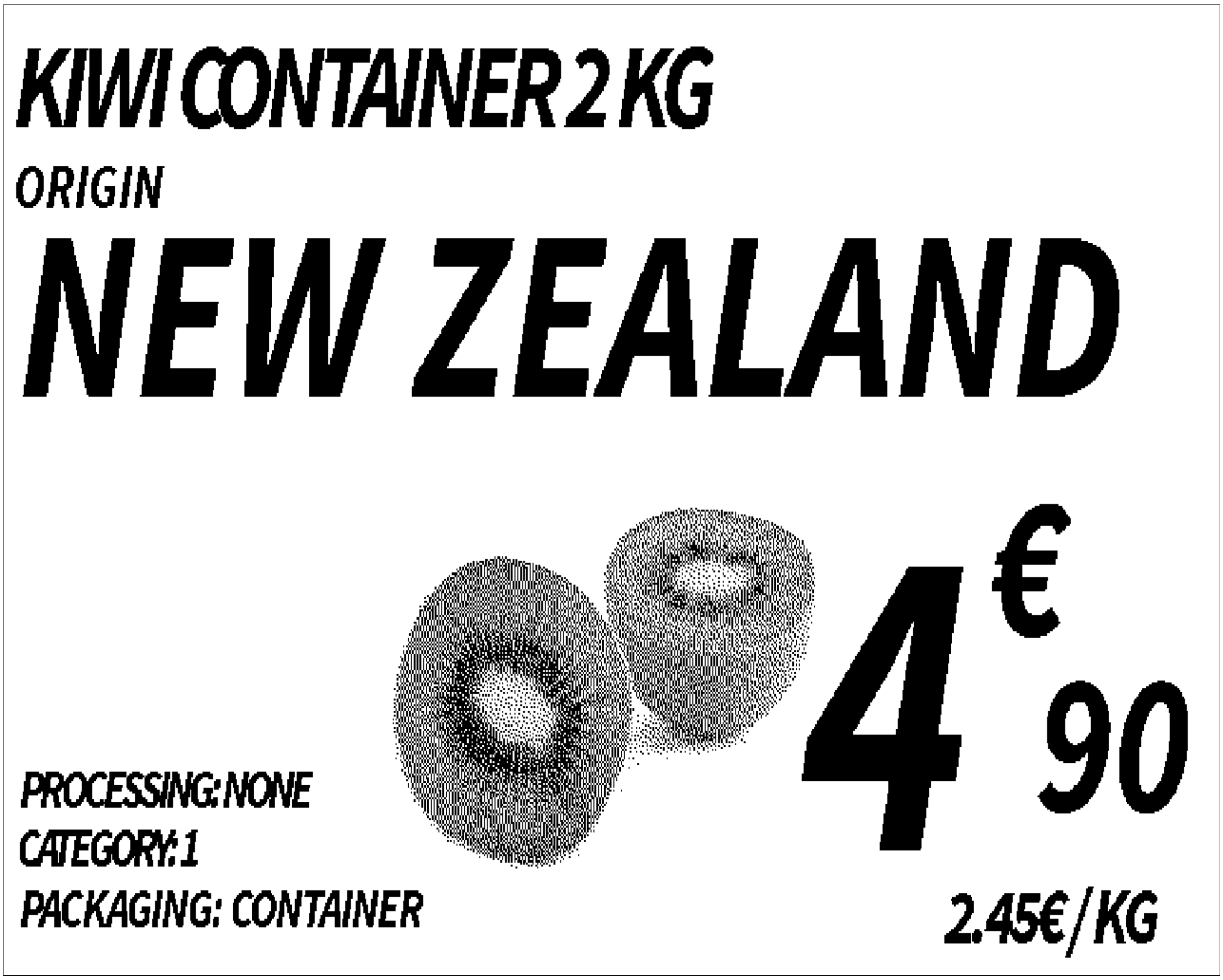
123456789012

Figure 3.1: 4.41″ ESL test image



KIWI CONTAINER 2 KG
ORIGIN
NEW ZEALAND
PROCESSING: NONE
CATEGORY: 1
PACKAGING: CONTAINER
4 € 90
2.45€/KG

Figure 3.2: 7.4″ ESL test image



KIWI CONTAINER 2 KG
ORIGIN
NEW ZEALAND
PROCESSING: NONE
CATEGORY: 1
PACKAGING: CONTAINER
4 € 90
2.45€/KG

Figure 3.3: 10.2″ ESL test image

3.5 DC Characteristics

Symbol	Description	Min	Max	Unit
VIH	Input high level voltage	0.7×VDD	–	V
VIL	Input low level voltage	–	0.3×VDD	V
VOH	Output high level voltage	VDD-0.4	–	V
VOL	Output low level voltage	–	0.4	V

Table 3.11: Typical operating conditions

4 Display Refresh Time

The T_{amb} temperature value indicates the middle of the range.

Example: T_{amb}=20 indicates range between 17.5 and 22.5°C.

4.1 Default Transition

T_{amb} [°C]		0	5	10	15	20	25	30	35	40	45	50
Time [s]	**TCS2-P441-231**	9.7	7.3	5.5	4.2	3	2.7	2.1	2.1	2.1	2.1	2.1
	TCS2-P74-231	8	6.8	5.7	4.5	3.3	3	2.6	2.6	2.2	2.2	2.2
	TCS2-P102-231	6.8	5.9	4.9	3.7	2.7	2.1	2.1	2.1	2.1	1.8	1.8

Table 4.1: Display refresh time versus ambient temperature – Default transition

4.2 Flashless Transition

T_{amb} [°C]		0	5	10	15	20	25	30	35	40	45	50
Time [s]	**TCS2-P441-231**	3.5	2.7	1.9	1.5	1.3	1.2	1.2	1.0	1.0	1.0	1.0
	TCS2-P74-231	2.5	2.3	1.9	1.8	1.6	1.5	1.3	1.3	1.3	1.1	1.1
	TCS2-P102-231	2.4	2.1	1.8	1.6	1.3	1.0	1.1	1.1	1.1	1.1	1.1

Table 4.2: Display refresh time versus ambient temperature – Flashless transition

4.3 Flashless-Inverted Transition

T_{amb} [°C]		0	5	10	15	20	25	30	35	40	45	50
Time [s]	**TCS2-P441-231**	6.3	4.8	3.3	2.5	2.1	1.9	1.7	1.5	1.5	1.5	1.5
	TCS2-P74-231	4.3	3.8	3.1	2.8	2.5	2.2	1.9	1.9	1.9	1.6	1.6
	TCS2-P102-231	4.1	3.6	3.1	2.6	2.1	1.5	1.6	1.6	1.6	1.6	1.6

Table 4.3: Display refresh time versus ambient temperature – Flashless-Inverted transition

5 TCS2 Hands-on

Unless specified otherwise, the values in this chapter are applicable to the whole TCS2 product family, i.e. to both the TC2 and TCM2.

5.1 TC2 Integration

TC2 together with the reference schematic can be integrated with user's own host system. This enables the user to develop their own application utilizing ePaper technology.

Reference design is included in the Design Guide, distributed separately. Please contact sales@mpicosys.com for more information.[1)]

5.2 TCM2 Interconnection

Use the below described host connector to connect TCM2 to your host system. The host connector is a 10-pin single-row 2.54 mm-pitch male header.

NOTE Forward slash "/" in front of the pin name indicates the signal is active low

Pin #	Pin Name	Remarks
1	GND	Supply ground
2	/TC_EN	TC2 enable
3	VDDIN[2)]	Power supply for digital part[2)]
4	VIN[2)]	Power supply for analog part[2)]
5	/TC_BUSY	Host interface busy output
6	TC_MISO	Host interface data output
7	TC_MOSI	Host interface data input
8	/TC_CS	Host interface chip select input
9	TC_SCK	Host interface clock input
10	GND	Supply ground

Table 5.1: TCM2 host connector pinout

5.3 TCM2 Power On

Connect your power supply to the VDDIN and VIN pins.

VDDIN supply for digital part has to be supplied from a stable power supply, e.g. stabilized by a DC/DC converter or a low-dropout regulator (LDO).

In TCS2-P441-231, VIN is internally connected to VDDIN, so this pin may be left not connected.

In TCS2-P74-231 and TCS2-P102-231, VIN can either be supplied directly from the battery (e.g. coin-cell) for improved efficiency, or can be shorted to VDDIN.

1) TC2 for 10.2″ display is not available
2) Due to the fact TCS2-P441-231 utilizes internal display panel charge pump, the supply pins VDDIN and VIN are shorted with a 0R resistor.

When connected to power supply, TCS2 is by default turned off to conserve energy. To switch it on, activate the /TC_EN signal.

5.4 Interface

Connection To Host

User's host system can communicate with TCS2 via Serial Peripheral Interface (SPI) with additional /TC_EN and /TC_BUSY line. TCS2 works as an SPI slave device. TCS2 power has to be supplied by the host system. The SPI supports 8-bit frames of data flowing from the master to the slave and from the slave to the master.

In case of bus operation when more devices operate on the same SPI interface, signals TC_SCK, TC_MISO, TC_MOSI, /TC_CS, /TC_BUSY can be shared with other SPI devices. In typical use, each device has assigned unique CS signal. If more than one TCS2 (with the same device system) is connected to one /TC_CS line, the TCS2s work synchronously executing the same commands. In this configuration there is a possible electrical conflict on TC_MISO line and response data might be unreliable.

Signals

Inputs:

- /TC_EN – active low; enables the VDD for the TCS2 MCU
- /TC_CS – active low
  - low level: activates SPI as slave
  - high level: deactivates SPI
  - event on rising edge – data analysis
- TC_SCK – SPI clock input
- TC_MOSI – SSP0 peripheral data input, no pulls
- Outputs:
- TC_MISO
  - when /TC_CS is active: SSP0 peripheral data output, no pulls
  - when /TC_CS is inactive: input GPIO with internal pull-down
- /TC_BUSY – active low; GPIO pseudo open-drain[1];
  - low level: GPIO output, no pulls
  - high level: GPIO input with internal pull-up (min. 20kR, typ. 50kR)

Startup and Initialization Sequence

TCS2-P441-231:

The below timing diagram (Figure 5.1) represents the startup and initialization sequence after power-up. The TCS2-P441-231 is ready for communication after $T_{STARTUP}$ + T_{INIT} which is indicated by /TC_BUSY rising edge.

[1] It performs a logical AND function in case of many TC sharing the same BUSY line.

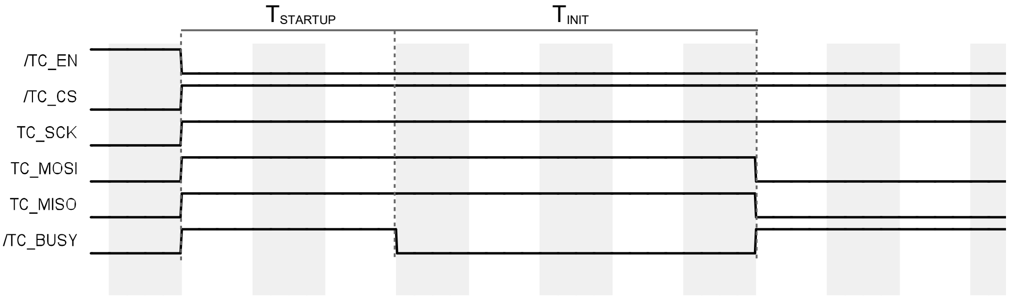


Figure 5.1: TCS2-P441-231 initialization sequence

Time	Min	Max	Unit
$T_{STARTUP}$	1.5	1.5	ms
T_{INIT}	4.2	70	ms

Table 5.2: TCS2-P441-231 startup and initialization times

TCS2-P74-231:

The below timing diagram (Figure 5.2) represents the startup and initialization sequence after power-up. The TCS2-P74-231 is ready for communication after $T_{STARTUP}$ + T_{INIT} which is indicated by /TC_BUSY rising edge.

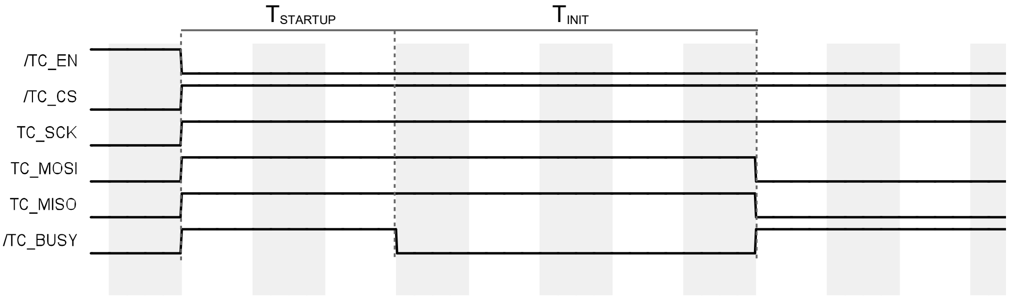


Figure 5.2: TCS2-P74-231 initialization sequence

Time	Min	Max	Unit
$T_{STARTUP}$	-	4.2	ms
T_{INIT}	1.5	70	ms

Table 5.3: TCS2-P74-231 startup and initialization times

TCS2-P102-231:

The below timing diagram (Figure 5.3) represents the startup and initialization sequence after power-up. The TCS2-P102-231 is ready for communication after $T_{STARTUP+INIT}$ which is indicated by /TC_BUSY rising edge.

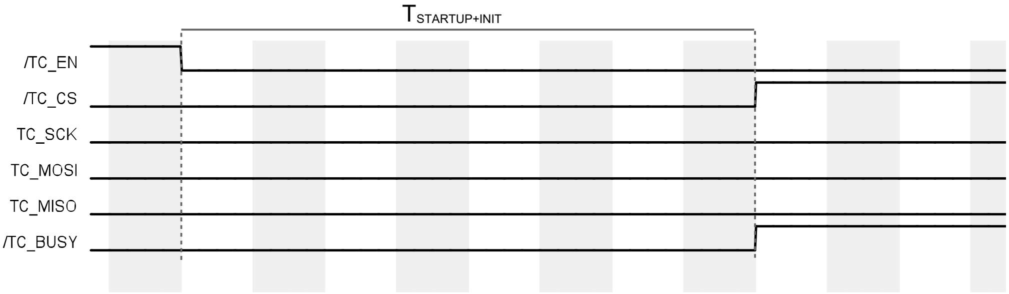


Figure 5.3: TCS2-P102-231 initialization sequence

Time	Min	Max	Unit
$T_{STARTUP+INIT}$	2.2	33	ms

Table 5.4: TCS2-P102-231 startup and initialization time

SPI Settings

- Bit rate – up to 12 MHz
    - Effective bit rate: 3 MHz
- Polarity – CPOL = 1; clock transition high-to-low on the leading edge and low-to-high on the trailing edge
- Phase – CPHA = 1; setup on the leading edge and sample on the trailing edge
- Bit order – MSB first
- Chip select polarity – active low

Reference SPI timing diagram below:

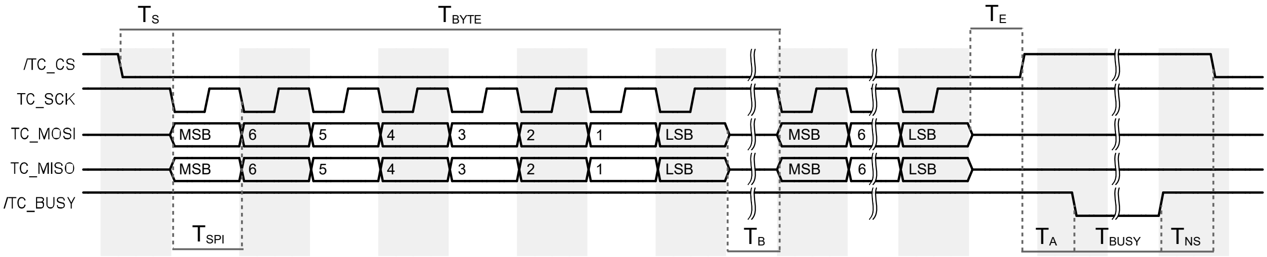


Figure 5.4: SPI timing diagram

Time	T_S	T_{BYTE}	T_{SPI}	T_B	T_E	T_A	T_{BUSY}	T_{NS}
Min.	2.67 µs	2.67 µs	83 ns[1]	0[2]	2.4 µs	–	255 µs	50 ns
Max.	–	–	550 ns	–	–	14.5 µs	–	–

Table 5.5: TCS2-P441-231_v1.1 SPI timing description

1) Minimum T_{SPI} value reflects the maximum supported bit rate of 12 MHz. In this case T_B has to be greater than or equal to 2.0 µs.
2) T_{SPI} value can equal 0 if frequency is lower than or equal to 3 MHz.

Time	T_S	T_{BYTE}	T_{SPI}	T_B	T_E	T_A	T_{BUSY}	T_{NS}
Min.	2.67 µs	2.22 µs	80 ns[1]	0[2]	2.4 µs	–	255 µs	50 ns
Max.	–	–	550 ns	–	–	14.5 µs	–	–

Table 5.6: TCS2-P74-231_v1.0 SPI timing description

Time	T_S	T_{BYTE}	T_{SPI}	T_B	T_E	T_A	T_{BUSY}	T_{NS}
Min.	50 ns	855 ns	83 ns[3]	94 ns[4]	50 ns	–	12.4 µs	–
Max.	–	–	1 ms	–	–	3.8 µs	–	–

Table 5.7: TCS2-P102-231_v1.2 SPI timing description

Communication Flow

TCon is able to communicate to the host system if /TC_BUSY signal is inactive. To start communication, the /TC_CS line has to be activated by the host. Then the command data can be passed. There is no timeout during the communication, so delays between the consecutive bytes are allowed. The command is interpreted by the TCS2 only after /TC_CS line has been deactivated.

After passing the command, it is being interpreted and executed by the TCS2. The time of execution is indicated by /TC_BUSY signal active. During this time, the TCS2 does not accept any new commands.

5.5 Framebuffer Slots

TCS2 features image (framebuffers) slots for storing the image data. The main purpose of enabling multiple slots is to increase the product use cycles, limited by the flash memory endurance. With multiple slots, the newly uploaded images (when addressed with slot number 0) are cycling through the slots, effectively reducing the necessity of constantly erasing the same memory sectors. The images are stored in non-volatile memory, thus are retained when the system is not powered. The number of the available framebuffer slots is indicated in the Table 5.8 below. The default 8 Mbit memory can be replaced with a more sizable memory chip, allowing greater slots number, up to 99 with a 128 Mbit memory in TCS2-P102-231.

TCS Type	Default number of slots	Maximum number of slots
TCS2-P441-231	16	32
TCS2-P74-231	16	32
TCS2-P102-231	3	99

Table 5.8: Number of available image slots

When addressed with the default slot number 0, TCS2 automatically assigns the slot number so that the new data is always stored in the oldest used slot, and correctly referred to in case of display update command. In this use case the host does not need to trace in which slot the data is stored.

Framebuffer slots can also be addressed directly by the slot number (1, 2, 3, …) or relatively by the last displayed: *-1* addresses the currently displayed slot, *-2* addresses the previously displayed slot, and so on. In this case the host needs to trace which slot the data has been written to and which slot is

[1] Minimum T_{SPI} value reflects the maximum supported bit rate of 12 MHz. In this case T_B has to be greater than or equal to 2.0 µs.
[2] T_{SPI} value can equal 0 if frequency is lower than or equal to 3 MHz.
[3] Minimum T_{SPI} value reflects the maximum supported bit rate of 12 MHz. In this case T_B has to be greater than or equal to 2.0 µs.
[4] T_{SPI} value can equal 0 if frequency is lower than or equal to 3 MHz.

available for upload. It is important to note that the slot containing the data of the currently displayed image cannot be overwritten.

5.5.1 Current Slot Restriction

The slot that has been displayed latest, cannot be modified. This is because the last displayed image data is needed to properly refresh the display, minimizing the ghosting phenomenon.

Thus all the commands attempting to modify the last displayed slot will respond with 0x6981 error status code.

5.6 Use Cases

5.6.1 Regular Image Upload and Display

The most common use case of the TCS2 product assumes uploading and displaying an image. The most basic use case utilizes automatic slot distribution, which relieves user from controlling how the EPD image date is stored in TCS2 memory.

Example use case:

1) Evoke 5.7.1.1 UploadImageData command with argument P2=0 to fill one of the framebuffer slots with image data
2) Evoke 5.7.2.1 DisplayUpdate command with argument P2=0 to display uploaded image
3) After filling all available framebuffer slots with images, the sequence of commands upload-display will continue to work, since automatic slot distribution will replace unused slots with new images

5.6.2 Partial Image Upload and Display

The partial upload and display allows to:

- Define any rectangular area within the display (Region of Interest – ROI)
- Fill the area with specific data, or
- Fill the area with data from another, previously populated image slot, or
- Fill the area with uniform data (black, white, or pattern)

The commands can be run in a sequence to generate the image, and then display the final result.

NOTE The generated image can be displayed either with full-quality display refresh (with black and white flashes), or with flashless update. Using the flashless update will ensure that only the changed part of the display is refreshed – see 5.7.2.1 DisplayUpdate.

Each image building sequence needs to be proceeded by memory slot erasing.

With the use of partial update commands described below, user can e.g. upload part of an image to the buffer, copy an image from another slot and fill the rest of the image with black, white or with given pattern.

Example use case:

1) Erase chosen slot (see 5.7.1.5 ImageEraseFrameBuffer) – erase is recommended before any partial upload operation

2) Evoke 5.7.1.6 ImageUploadSetROI command to define the area (e.g. 100x200 px in the upper-right corner) that is about to be modified
3) Evoke 5.7.1.1 UploadImageData commands to fill the above-defined area with image data
4) Evoke 5.7.1.6 ImageUploadSetROI command to define the next area (areas cannot intersect)
5) Use 5.7.1.8 ImageUploadCopySlots command to copy image data from another slot to the defined area
6) Use 5.7.1.6 ImageUploadSetROI command to define the next area (areas cannot intersect)
7) Use 5.7.1.7 ImageUploadFixVal to e.g. fill the defined area with white color
8) Evoke 5.7.2.1 DisplayUpdate command to display created image
9) The sequence of commands (2-3, 4-5, 6-7) can be run in any order

5.7 Command Description

Command Format

Each command is built up from 3 to 255 bytes. The command is divided into six fields.

The first three fields are used in every command:

- *INS* – command group specific
- *P1* – parameter
- *P2* – parameter

whereas the next three fields are only used by some particular commands:

- *Lc* – number of bytes in *Data* field
- *Data* – bytes forming command data; number of bytes determined by *Lc*
- *Le* – number of bytes of expected response

Response

Upon each command, TCS2 responds with a 2-byte command status code. The command status code is not included in the *Le* (expected response length) count.

Status codes list:

Status code	Status mnemonic	Description
0x9000	EP_SW_NORMAL_PROCESSING	Command executed successfully
0x6581	EP_SW_MEMORY_FAILURE	An error occurred while interfacing external memory
0x6700	EP_SW_WRONG_LENGTH	Incorrect length (invalid *Lc* value or command too short or too long)
0x6981	EP_FRAMEBUFFER_SLOT_NOT_AVAILABLE	Frambuffer slot number is either the last displayed slot, or the number is out of range
0x6A00	EP_SW_WRONG_PARAMETERS_P1P2	Invalid *P1* or *P2* field
0x6A84	EP_FRAMEBUFFER_SLOT_OVERRUN	Framebuffer slot overridden
0x6C00	EP_SW_INVALID_LE	Specified value for *Le* field is invalid
0x6D00	EP_SW_INSTRUCTION_NOT_SUPPORTED	Command not supported
0x6F00	EP_SW_GENERAL_ERROR	Internal TCS2 MCU reset triggered due to abnormal behavior; the command was not executed properly
0x9D54	EP_DISPLAY_NOT_AVAILABLE	Display drivers are broken or display is not attached (available only in TCS2-P441-231)

Figure 5.5: List of response codes

If a command returns specific data, the status code is appended to the end of the data.

Data Readout

During each SPI clock cycle a full-duplex data transmission takes place: the host sends a bit on the MOSI line, and the TCS2 sends a bit on the MISO line at the same time.

Thus, the command status should be read after the command is executed. To read the command status, the host should send the expected number of 0x00 bytes to TCS2. The amount of bytes to be sent is dependent command type:

- If the command does not use the *Le* field, it will return only the two-byte status code; thus only two bytes should be sent by the host
- When *Le* field is used and set to 0x00, the response length is not determined; then the response should be read until 0x00 is encountered, indicating the response termination, and two additional bytes should be sent to acquire the command status
- When *Le* field is set to a value other than 0x00, the response length is determined by the value at *Le* field. The host should send the number of bytes indicated by the *Le* field, and two additional bytes to acquire the command status

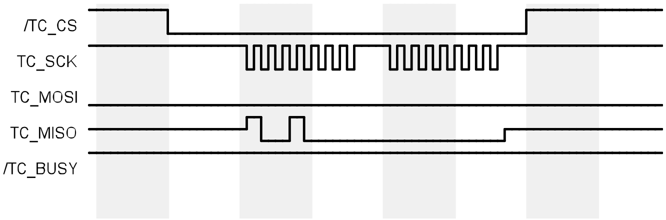


Figure 5.6: Example readout - 0x9000 response

5.7.1 Image Data Commands

This group of commands, handles the process of data transfer between the host and the TCS2.

5.7.1.1 UploadImageData

Command

INS	P1	P2	Lc	Data
0x20	0x01	Si	Data packet size (max 0xFA)	[Lc Data bytes]

Description

The command uploads image data (in EPD file format) to TCS2 image memory. The data needs to be divided into packets and transferred with multiple UploadImageData commands. In order to send the full image data, the user has to make sure to send it packet by packet.

While writing to the TCS2 internal memory, the TCS2 data pointer is internally increased by the size of the current packet, until reaching maximum of slot memory. When the slot memory size is exceeded, EP_FRAMEBUFFER_SLOT_OVERRUN status code will be returned as response.

Regardless of the uploaded EPD pixel data format type, the image data is automatically converted and stored in the TCS2 memory in certain Pixel Data Format Type:

- Type 2 in TCS2-P441-231_v1.1

- Type 4 in TCS2-P74-231_v1.0
- Type 0 in TCS2-P102-231_v1.2

Parameters

- P2: Si – framebuffer slot number (see 5.5 Framebuffer Slots)

Data

Image file in EPD format, see 6 EPD File Format). Maximum packet size is 251 bytes (as maximum command size is 255 bytes.)

NOTE If this command is used in partial update (i.e. following the ImageUploadSetROI command), the data **should not** contain the EPD header, and should be encoded in EPD format type 0.

Response

- 2-byte status code

5.7.1.2 GetImageData

Command

INS	P1	P2	Le
0xA0	0x01	Si	Length

Description

Get image data from specified slot. The data is divided into packets – similarly to UploadImageData command. To get the full image data, multiple GetImageData commands need to be sent, until the full image is received.

Each time the command is called, the TCS2 data pointer is increased by the size of the read data packet, until reaching the maximum of the slot size.

Parameters

- P1: Constant value
- P2: Si – framebuffer slot number (see 5.5 Framebuffer Slots)

Data

Image file in EPD format (see 6 EPD File Format.) Maximum packet size is 251 bytes (as maximum command size is 255 bytes.)

Response

- 2-byte status code

5.7.1.3 GetChecksum

Command

INS	P1	P2	Le
0x2E	0x01	Si	0x02

Description

Get 16-bit checksum of an image stored in the TCon memory.

NOTE Checksum is calculated based on Pixel Data Format Type used to store images. Images are stored in Type 2 in TCS2-P441-231_v1.1, in Type 4 in TCS2-P74-231_v1.0 and in Type 0 in TCS2-P102-231_v1.2.

Initial checksum value is 0x6363. Checksum is calculated on raw image data (including EPD header, see 6.1 Header.)

Checksum implementation:

```
uint16_t crc16_add(uint8_t byte, uint16_t acc)
{
   acc ^= byte;
   acc  = (acc >> 8) | (acc << 8);
   acc ^= (acc & 0xff00) << 4;
   acc ^= (acc >> 8) >> 4;
   acc ^= (acc & 0xff00) >> 5;
   acc = acc;
   return acc;
}
```

Parameters

- P1: Constant value
- P2: Si – framebuffer slot number (see 5.5 Framebuffer Slots)

Response

- [2 bytes: (0xHH, 0xLL), where 0xHH is the upper byte, and 0xLL is the lower byte of the 16 bit checksum.] + 0x9000 status code, or
- 2-byte error status code

5.7.1.4 ResetDataPointer

Command

INS	P1	P2
0x20	0x0D	0x00

Description

The command resets data pointer for Upload Image Data command.

NOTE Data pointer is automatically reset when TCS2 is enabled by /TC_EN activation

Parameters

- P1, P2: Constant values

Response

- 2-byte status code

5.7.1.5 ImageEraseFrameBuffer

Command

INS	P1	P2
0x20	0x0E	Si

Description

This command resets data pointer to the beginning of the chosen image slot (similarly to ResetDataPointer command) and erases the entire image slot. The erased slot is filled with 0xFF, which if displayed is shown as a full-black image.

Parameters

- P1: Constant value
- P2: Si – framebuffer slot number (see 5.5 Framebuffer Slots)

Response

- 2-byte status code

5.7.1.6 ImageUploadSetROI

Command

INS	P1	P2	Lc	Data
0x20	0x0A	0x00 or Si	0x08	[ROI data]

Description

This command sets region of interest for image upload. The framebuffer pointer is set to the beginning of ROI buffer: after the command each UploadImageData command will fill framebuffer in ROI region only.

- EPD image data header should not be sent in image data after ImageUploadSetROI command
- Set ROI region is valid until ResetDataPointer, ImageEraseFrameBuffer, DisplayUpdate or next ImageUploadSetROI command
- X coordinate value for ROI must be dividable by 8
- At startup and after ResetDataPointer, ImageEraseFrameBuffer and DisplayUpdate command ROI is not set and points to the whole framebuffer

Parameters

- P2: Constant value in TCS2-P441-231 and TCS2-P74-231 or Si – framebuffer slot number (see 5.5 Framebuffer Slots) in TCS2-P102-231

Data

ROI_data: four 16-bit (MSB first) values that define ROI area: Xmin (inclusive), Xmax (exclusive) (from 0 to 1024), Ymin (inclusive), Ymax (exclusive) (from 0 to 1280). Max has to be greater than min value.

Example: [01 C0 02 40 01 EC 03 14] defines ROI: Xmin = 448, Xmax = 576, Ymin = 492, Ymax = 788, which can fit 128x296 px image.

Response

- 2-byte status code

5.7.1.7 ImageUploadFixVal

Command

INS	P1	P2	Lc	Data
0x20	0x0B	Si	Length	[Pattern]

Description

This command copies and replicates the given data buffer (max 250 bytes specified by Data field, without EPD header) to frame buffer slot Si area specified by ROI which was set by ImageUploadSetROI. Can be used to clear framebuffer to white (Data = 0x00), black (Data = 0xFF) or pattern defined by Data. Framebuffer slot needs to be erased prior to partial image upload commands. Example use case of this command is barcode rendering.

Parameters

- P1: Constant value
- P2: Si – framebuffer slot number (see 5.5 Framebuffer Slots)

Data

Pattern data in EPD format type 0 (see 6 EPD File Format), without the EPD header. Maximum pattern size is 250 bytes.

Response

- 2-byte status code

5.7.1.8 ImageUploadCopySlots

Command

INS	P1	P2	Lc	Data
0x20	0x0C	Si	0x01	Si_Source

Description

This command copies image from a selected slot to framebuffer. If ROI was specified prior to this command, only this area is copied. ROI can be set by ImageUploadSetROI command. Otherwise the whole slot is copied.

Parameters

- P1: Constant value
- P2: Si – destination framebuffer slot number (see 5.5 Framebuffer Slots)

Data

- Si_Source – source framebuffer slot number (see 5.5 Framebuffer Slots)

Response

- 2-byte status code

5.7.2 Display Control Commands

5.7.2.1 DisplayUpdate

Command

INS	P1	P2	Lc	Data
Transition	0x01	Si	0x01	Temp

Description

The command starts the display refresh sequence, displaying the current content of the image memory, using the display transition sequence chosen with the INS value.

- If data was uploaded, the new data is going to be displayed
- If no data was sent, currently visible image will be refreshed (cleared and displayed again)

Parameters

- INS – Specifies the display refresh transition sequence according to the table below:

Transition	Name	Description
0x24, 0x82 (interchangeably)	Default	Default transition sequence – with black-white-black screen flashing. Offers the best image quality.
0x85	Flashless	Direct image to image transition (without the blank black or white screen in between.) The fastest and the most energy-efficient transition, at the cost of image quality.
0x86	Flashless-Inverted	Transition from the current image to the inverted new image, followed by the new image (without the blank black or white screen in between). Compromise between the Default and the Flashless – both in terms of energy consumption and image quality.

- P2: Si – slot number (see 5.5 Framebuffer Slots)

Data

- Temp (optional) – one byte defining temperature in degrees Celsius, U2 encoded, used for temperature compensation substituting the actual temperature readings; available only in TCS2-P102-231_v1.2
- Response
- 2-byte status code

5.7.2.2 EnableBlockDriving

Command

INS	P1	P2
0x22	0x01	mode

Description

The command controls the Block Driving display driving method, which is turned off by default in TCS2-P102-231_v1.2 (see 1.4.1 Block Driving). Command available only in TCS2-P102-231_v1.2.

Parameters

- P1: Constant value
- P2: mode – 0x01 to enable block driving or 0x00 to disable block driving

Response

- 2-byte status code

5.7.2.3 SetSlotsNumber

Command

INS	P1	P2
0x29	Si	0x00

Description

The command changes the maximum number of framebuffer slots (see 5.5 Framebuffer Slots) available for user. Information about current maximum number of slots is stored in nonvolatile memory. Note that more slots require more memory space. Tcon does not check if specified maximum slot number can be utilized with current memory, therefore the command must be used consciously.

Parameters

- P1: Si – maximum number of slots
- P2: Constant value

Response

- 2-byte status code

5.7.3 Device Info Commands

This group of commands, starting with INS = 0x30 byte, manages the acquirement of hardware information from TCon.

5.7.3.1 GetDeviceInfo

Command

INS	P1	P2	Le
0x30	0x01	0x01	0x00

Description

The command returns information on system hardware. String data is specific for the particular device type and is constant for the same type of devices if no hardware differences occur.

Parameters

- P1, P2: Constant values

Response

- [String: “MpicoSys TC2-P441-231_v1.1” terminated by 0x00 byte] + 0x9000 status code in case of TC2-P441-231_v1.1, or
- [String: “MpicoSys TC2-P74-231_v1.0” terminated by 0x00 byte] + 0x9000 status code in case of TC2-P74-231_v1.0, or

- [String: “MpicoSys TC2-P102-231_v1.2” terminated by 0x00 byte] + 0x9000 status code in case of TC2-P102-231_v1.2, or
- 2-byte error status code

5.7.3.2 GetDeviceId

Command

INS	P1	P2	Le
0x30	0x02	0x01	0x14

Description

The command returns unique device ID number.

Parameters

- P1, P2: Constant values

Response

- [20 bytes of data] + 0x9000 status code, or
- 2-byte error status code

5.7.4 System Info Commands

This group of commands, starting with INS = 0x31 byte, deals with acquirement of firmware information from TCon.

5.7.4.1 GetSystemInfo

Command

INS	P1	P2	Le
0x31	0x01	0x01	0x00

Description

The command returns information on system firmware.

Parameters

- P1, P2: Constant values

Response

- [String: “MpicoSys TC2-P441-231_fB_BIN” terminated by 0x00 byte] + 0x9000 status code in case of TC2-P441-231_v1.1, or
- [String: “MpicoSys TC2-P74-231_fA_BIN” terminated by 0x00 byte] + 0x9000 status code in case of TC2-P74-231_v1.0, or
- [String: “MpicoSys TC2-P102-231_fC_BIN” terminated by 0x00 byte] + 0x9000 status code in case of TC2-P102-231_v1.2, or
- 2-byte error status code

5.7.4.2 GetSystemVersionCode

Command

INS	P1	P2	Le
0x31	0x02	0x01	0x10

Description

The command returns information on system version.

Parameters

- P1, P2: Constant values

Response

- 0x D0 AE 01 00 00 00 00 00 33 10 04 00 00 00 00 00 + 0x9000 status code in case of TCS2-P441-231_v1.1, or
- 0x D0 B1 00 00 00 00 00 00 3A 10 04 00 00 00 00 00 + 0x9000 status code in case of TCS2-P74-231_v1.0, or
- 0x D0 AF 02 00 00 00 00 00 3D 20 04 00 00 00 00 00 + 0x9000 status code in case of TCS2-P102-231_v1.2, or
- 2-byte error status code

5.7.5 Sensor Data Commands

5.7.5.1 GetSensorData

Command

INS	P1	P2	Le
0xE5	0x01 or 0x04	0x00	0x02

Description

This command returns the temperature value measured by the TCS2 temperature sensor. The sensor is built in the TCM2 board and is included in the TCon reference design. The measurement is based on a NCP18WB473E03RB thermistor and 8-bit ADC.

Parameters

- P1: either 0x01 for raw ADC data or 0x04 for degrees Celsius, U2 encoded data
- P2: Constant value

Response

- [2 bytes of temperature measurement in deg. C, U2 encoded] + 0x9000 status code, or
- [2 bytes of RAW sensor data] + 0x9000 status code, or
- 2-byte error status code

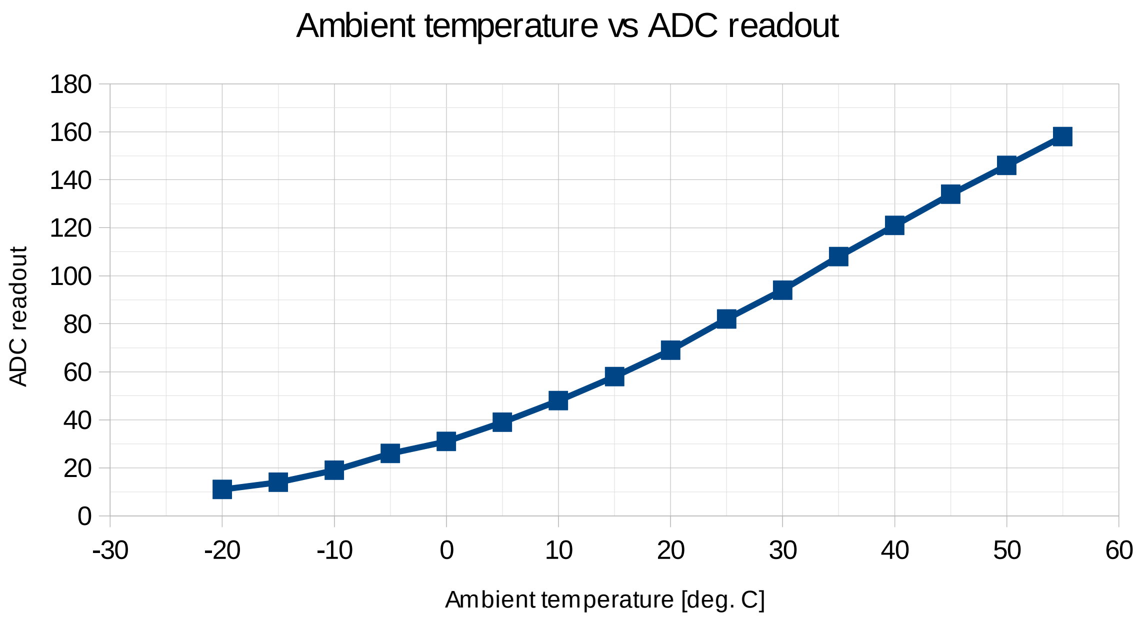


Figure 5.7: Ambient temperature versus ADC readout chart

Temperature [°C]	ADC value [dec]	ADC value [hex]
-20	11	0B
-15	14	0E
-10	19	13
-5	26	1A
0	31	1F
5	39	27
10	48	30
15	58	3A

Temperature [°C]	ADC value [dec]	ADC value [hex]
20	69	45
25	82	52
30	94	5E
35	108	6C
40	121	79
45	134	86
50	146	92
55	158	9E

6 EPD File Format

EPD is a specific raster graphics image file format, accepted by TCS2. EPD file format was developed to maximize the decoding efficiency on the target platform. The EPD file comprises of two parts:

- Header
- Image data

Table below describes the various panels resolution and corresponding image data array sizes, as well as EPD files sizes.

Panel size	Image resolution [px]	Image color depth [bit]	Header size [bytes]	Image data array size [bytes]	EPD file size [bytes]
4.41″	400×300	1	16	15,000	15,016
7.4″	480×800			48,000	48,016
10.2″	1024×1280			163,840	163,856

Table 6.1: TCS2 panels and corresponding image data

6.1 Header

EPD file begins with a header. The header size is 16 bytes. The consecutive bytes are described in the table below:

Field name	Size	Possible values	Description	
panel type	1 byte	0x33	Panel code	4.41″
				7.4″
		0x3D		10.2″
X res	2 bytes	0x0190	400 px	
		0x01E0	480 px	
		0x0400	1024 px	
Y res	2 bytes	0x012C	300 px	
		0x0320	800 px	
		0x0500	1280 px	
color depth	1 byte	0x01	Image color depth – 1-bit (black and white)	
Pixel Data Format Type	1 byte	0x00	Image pixel data format type 0	
		0x02	Image pixel data format type 2	
		0x04	Image pixel data format type 4	
RFU	9 bytes	0x00	Reserved for future use	

Table 6.2: EPD header breakdown

Based on the information from the table above, here are complete header values depending on the panel size, for image pixel data format 0:

- TCS2-P441-231: 0x 33 01 90 01 2C 01 00 00 00 00 00 00 00 00 00 00
- TCS2-P74-231: 0x 3A 01 E0 03 20 01 03 00 00 00 00 00 00 00 00 00
- TCS2-P102-231: 0x 3D 04 00 05 00 01 00 00 00 00 00 00 00 00 00 00

6.2 Image Data

Each byte of the image data encodes information on eight pixels (a single pixel is described by one bit of a single byte).

1-bit gray scale provides 2 colors. Bit value 0 corresponds to white color while value 1 represents black color.

6.2.1 Pixel Data Format Type 0

This format is used in TCS2-P441-231, TCS2-P74-231 and TCS2-P102-231. Each byte of image data shall convey information on 8 consecutive pixels of the RAW image.

Conversion Algorithm

The algorithm for conversion from standard RAW 4-bit data to EPD format is described below.

- Start with a byte array of image data which is already downsampled to 1-bit monochrome; each byte conveys information on 1 pixel

1) Get a single row of 8 bytes (8 pixels):

Input byte No.:	0	1	2	3	4	5	6	7
Pixel value:	0	1	1	1	0	1	1	0

Table 6.3: Input data – 8 bytes

2) Merge the input byte values (numbering from 0 to 7) into one output byte, conveying information on 8 pixels

Input byte No.:	0	1	2	3	4	5	6	7
Pixel value:	0	1	1	1	0	1	1	0
Output byte value:	0x76 (0b01110110)							

Table 6.4: Output data – single byte

3) Go back to Step 1), getting the following row; repeat until all the bytes are processed

Sample Code

Below is sample Java code for image conversion:

```
static byte[] convertTo1bit_PixelFormatType0(byte[] picData, int w, int h)
{
   byte[] newRow = new byte[picData.length * 1 / 8];
   // join nibbles (so 1 byte is 8 pixels)
   int j = 0;
   for (int i = 0; i < picData.length; i += 8)
   {
           newRow[j] = (byte)( ((picData[i + 0] << 7) & 0x80) |
                               ((picData[i + 1] << 6) & 0x40) |
                               ((picData[i + 2] << 5) & 0x20) |
                               ((picData[i + 3] << 4) & 0x10) |
                               ((picData[i + 4] << 3) & 0x08) |
                               ((picData[i + 5] << 2) & 0x04) |
                               ((picData[i + 6] << 1) & 0x02) |
                               ((picData[i + 7])      & 0x01));
           j++;
   }
   return newRow;
}
```

6.2.2 Pixel Data Format Type 2

This format is used in TCS2-P441-231, for backwards compatibility with Generation 1.

Conversion Algorithm

The algorithm for conversion from standard RAW 4-bit data to EPD format is described below.

- Start with a byte array of image data which is already downsampled to 1-bit monochrome; each byte conveys information on 1 pixel

1) Get a single row of 8 bytes (8 pixels):

Input byte No.:	0	1	2	3	4	5	6	7
Pixel value:	0	1	1	1	0	1	1	0

Table 6.5: Input data – 8 bytes

2) Assign the input byte values (numbering from 0 to 7) to the output byte, conveying information on 8 pixels, as follows:
   - Input byte 0: assign to output byte bit 0
   - Input byte 1: assign to output byte bit 2
   - Input byte 2: assign to output byte bit 4
   - Input byte 3: assign to output byte bit 6
   - Input byte 4: assign to output byte bit 1
   - Input byte 5: assign to output byte bit 3
   - Input byte 6: assign to output byte bit 5
   - Input byte 7: assign to output byte bit 7

Output byte bit No.:	0	1	2	3	4	5	6	7
Input byte No.:	0	4	1	5	2	6	3	7
Pixel value:	0	0	1	1	1	1	1	0
Output byte value:	0x3E (0b00111110)							

Table 6.6: Output data – single byte

3) Go back to Step 1), getting the following row; repeat until all the bytes are processed

Sample Code

Below is sample Java code for image conversion:

```
static byte[] convertTo1bit_PixelFormatType2(byte[] picData, int w, int h)
{
   byte[] newRow = new byte[picData.length * 1 / 8];
   // join nibbles (so 1 byte is 8 pixels) and interlace at the same time
   int j = 0;
   for (int i = 0; i < picData.length; i += 8)
   {
          newRow[j] = (byte)( ((picData[i + 0] << 7) & 0x80) |
                              ((picData[i + 4] << 6) & 0x40) |
                              ((picData[i + 1] << 5) & 0x20) |
                              ((picData[i + 5] << 4) & 0x10) |
                              ((picData[i + 2] << 3) & 0x08) |
                              ((picData[i + 6] << 2) & 0x04) |
                              ((picData[i + 3] << 1) & 0x02) |
                              ((picData[i + 7])      & 0x01));
          j++;
   }
   return newRow;
}
```

6.2.3 Pixel Data Format Type 4

This format is used in TCS2-P74-231, for backwards compatibility with Generation 1.

Conversion Algorithm

The algorithm for conversion from standard RAW 4-bit data to EPD format is described below.

- Start with a byte array of image data which is already downsampled to 1-bit monochrome; each byte conveys information on 1 pixel

1) Get a single row of 480 bytes (480 pixels) – the *Bytes value* represent the value of 8 consecutive bytes if merged into one byte:

Input Byte No.:	0÷7	8÷15	16÷23	24÷31	...	...	...	...	472÷479
Bytes value:	0x76	0x4C	0xA3	0x1F	...	...	...	...	...

Table 6.7: Input data – 60 bytes (480 pixels)

2) Get first 16 bytes from that row:

Input Byte No.:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pixel value:	0	1	1	1	0	1	1	0	0	1	0	0	1	1	0	0
Bytes value:	0x76 (0b01110110)								0x4C (0b01001100)							

Table 6.8: Input data – first 16 bytes

3) Create a 2-byte Intermediate array:

Int. byte No.:	0								1							
Position:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Input byte No.:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Pixel value:	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Int. byte value:	–								–							

Table 6.9: Intermediate array – empty

4) Assign the Input Byte values (numbering from 0 to 15) to Intermediate array, conveying information on 16 pixels, as follows:
   - Input byte 0: assign to Intermediate array position 6
   - Input byte 1: assign to Intermediate array position 8
   - Input byte 2: assign to Intermediate array position 4
   - Input byte 3: assign to Intermediate array position 10
   - Input byte 4: assign to Intermediate array position 2
   - Input byte 5: assign to Intermediate array position 12
   - Input byte 6: assign to Intermediate array position 0
   - Input byte 7: assign to Intermediate array position 14
   - Input byte 8: assign to Intermediate array position 7
   - Input byte 9: assign to Intermediate array position 9
   - Input byte 10: assign to Intermediate array position 5
   - Input byte 11: assign to Intermediate array position 11
   - Input byte 12: assign to Intermediate array position 3
   - Input byte 13: assign to Intermediate array position 13
   - Input byte 14: assign to Intermediate array position 1
   - Input byte 15: assign to Intermediate array position 15

Int. byte No.:	0								1							
Position:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Input Byte No.:	6	14	4	12	2	10	0	8	1	9	3	11	5	13	7	15
Pixel value:	1	0	0	1	1	0	0	0	1	1	1	0	1	1	0	0
Int. byte value:	0x98 (0b10011000)								0xEC (0b11101100)							

Table 6.10: Intermediate array – first 16 bytes processed

5) Create an output array of 60 bytes
6) Assign Intermediate byte 0 (the one with even pixels) to position 29 of the Output Array; Assign Intermediate byte 1 (the one with odd pixels) to position 59 of the Output Array

Output Byte No.:	0	1	...	...	...	...	28	29	30	31	...	...	...	...	58	59
Int. byte No.:	–	–	–	–	–	–	–	0	–	–	–	–	–	–	–	1
Output Byte value:	–	–	–	–	–	–	–	0x98	–	–	–	–	–	–	–	0xEC

Table 6.11: Output data filled in with first two bytes

7) Go back to Step 2), getting the following 16 bytes from the row (bytes 16÷31)

Input Byte No.:	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Bytes value:	0xA3 (0b10100011)								0x1F (0b00011111)							

Table 6.12: Input data – following 16 bytes

- Assign the bytes as in Step 4)

Int. byte No.:	2	3
Int. byte value:	0xDA (0b11011010)	0x17 (0b00010111)

Table 6.13: Intermediate data – following 16 bytes processed

- Assign Intermediate byte 2 to position 28 of the Output Array
  Assign Intermediate byte 3 to position 58 of the Output Array

Output Byte No.:	0	1	...	...	...	...	28	29	30	31	...	...	...	...	58	59
Int. byte No.:	–	–	–	–	–	–	2	0	–	–	–	–	–	–	3	1
Output Byte value:	–	–	–	–	–	–	0xDA	0x98	–	–	–	–	–	–	0x17	0xEC

Table 6.14: Output data filled in with following two bytes

8) Repeat until the full row is processed
9) Get the following row and repeat the process until the whole image has been processed

Sample Code

Below is sample Java code for image conversion:

```
static byte[] convertTo1bit_PixelFormatType4(byte[] picData, int w, int h)
{
   byte[] newPicData = new byte[picData.length / 8];
   int row = 30, s = 1;
   for (i = 0; i < picData.length; i += 16)
   {
           newPicData[row-s] = (byte)(       ((picData[i + 6 ] << 7) & 0x80) |
                                             ((picData[i + 14] << 6) & 0x40) |
                                             ((picData[i + 4 ] << 5) & 0x20) |
                                             ((picData[i + 12] << 4) & 0x10) |
                                             ((picData[i + 2 ] << 3) & 0x08) |
                                             ((picData[i + 10] << 2) & 0x04) |
                                             ((picData[i + 0 ] << 1) & 0x02) |
                                             ((picData[i + 8 ] << 0) & 0x01));

           newPicData[row+30-s] = (byte) (   ((picData[i + 1 ] << 7) & 0x80) |
                                             ((picData[i + 9 ] << 6) & 0x40) |
                                             ((picData[i + 3 ] << 5) & 0x20) |
                                             ((picData[i + 11] << 4) & 0x10) |
                                             ((picData[i + 5 ] << 3) & 0x08) |
                                             ((picData[i + 13] << 2) & 0x04) |
                                             ((picData[i + 7 ] << 1) & 0x02) |
                                             ((picData[i + 15] << 0) & 0x01));
           s++;
           if(s==31)
           {
                  s=1;
                  row+=60;
           }
   }
   return newPicData;
}
```

7 Revision History

Document Revision	Release Date	Document Status	Supersedes
E	2016-12-19	Approved	D
D	2016-12-19	Approved	C
C	2016-09-23	Approved	B
B	2015-11-23	Approved	A
A	2015-07-17	Approved	All the draft versions

Table 7.1: Revision history

Document Revision	Change Log
E	TCS2-P74 default slot number information corrected
D	Corrected GetImageData P1 value information
C	Update for TCS2-P74-231_v1.0 and TCS2-P102-231_v1.2
B	Update for v1.1 versions
A	Initial version

Table 7.2: Change log

8 Legal Information

Copyright

All trademarks, service marks and product or service names are trademarks or registered trademarks of their respective owners.

General

Information in this document is believed to be accurate and reliable. However, MpicoSys does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes

MpicoSys reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use

MpicoSys products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a MpicoSys product can reasonably be expected to result in personal injury, death or severe property or environmental damage. MpicoSys accepts no liability for inclusion and/or use of MpicoSys products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk. Product described in this document is intended for development purposes only and comes without any warranty. MpicoSys accepts no liability for inclusion and/or use of MpicoSys products in commercial products or applications and therefore such inclusion and/or use is at the customer's own risk.

Any software is provided "as is" and any expressed or implied warranties are disclaimed. In no event shall MpicoSys be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, arising in any way out of the use of the software.

Applications

Applications that are described herein for any of these products are for illustrative purposes only. MpicoSys makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Absolute maximum ratings

Stress above one or more limiting values of Absolute Maximum Ratings System (as defined in the Absolute Maximum Ratings System of IEC 60134) may cause permanent damage to the device. Limiting values are stress ratings only and operation of the device at these or any other conditions above those given in the Characteristics sections of this document is not implied. Exposure to limiting values for extended periods may affect device reliability.

Terms and conditions of sale

MpicoSys products are sold subject to the general terms and conditions of commercial sale, as published at http://www.mpicosys.com/terms, including those pertaining to warranty, intellectual property rights infringement and limitation of liability, unless explicitly otherwise agreed to in writing by MpicoSys. In case of any inconsistency or conflict between information in this document and such terms and conditions, the latter will prevail.

No offer to sell or license

Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

9 Contact Information

If you have any technical questions, please send an email to support@mpicosys.com.

Please contact sales@mpicosys.com for commercial information.