Developer's Guide

Timing Controller Solutions – E Ink Panels

TCM2-E97-320_v1.0
TCM2-E133-320_v1.3
TCM2-E312-320_v1.2

Classification**: Public**

Document Revision: **E**

Table of Contents

1 Introduction

ePaper **Timing Controller Solutions Generation 2** (TCS2) provide timing controller functionality for E Ink large size panels (**9.7”, 13.3” or 31.2”**). 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.

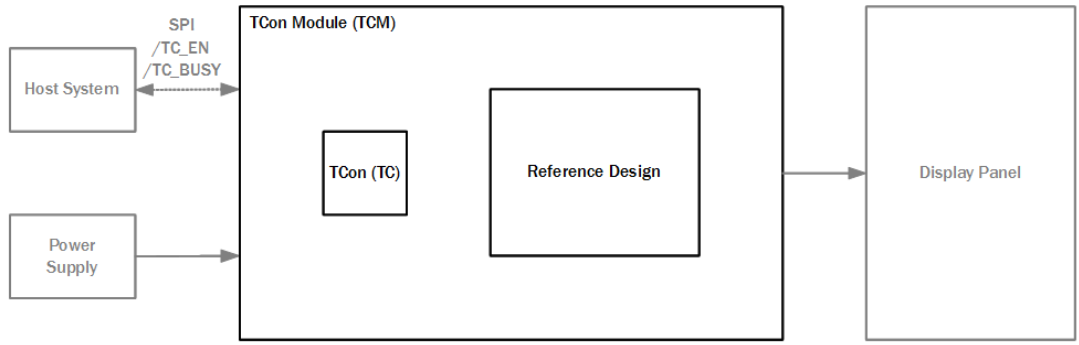


Figure 1.1: TCS block diagram

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 E Ink Displays (TCS2-E)

TCS2-E97-320 Product Codes

Timing Controller Module: **TCM2-E97-320_v1.0**

Timing Controller: **TC2-E97-320_v1.0**

TCS2-E133-320 Product Codes

Timing Controller Module: **TCM2-E133-320_v1.3**

Timing Controller: **TC2-E133-320_v1.3**

TCS2-E312-320 Product Codes

Timing Controller Module: **TCM2-E312-320_v1.2**

Timing Controller: **TC2-E312-320_v1.2**

1.2 Supported Display Panels

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

Please consult the respective E ink display *Technical 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	E Ink Display Size	E Ink Display Material (FPL)	E Ink Display Part No.	E Ink Display Resolution [px]	E Ink Display Density [dpi]
TCS2-E97-320_v1.0	9.7″	Carta – v320	ED097TC2	1200x825	150
TCS2-E133-320_v1.3	13.3″		ED133UT2	1600×1200	150
TCS2-E312-320_v1.2	31.2″		ED312TT2	2560×1440	94

Table 1.1: Supported display panels summary

1.3 Features

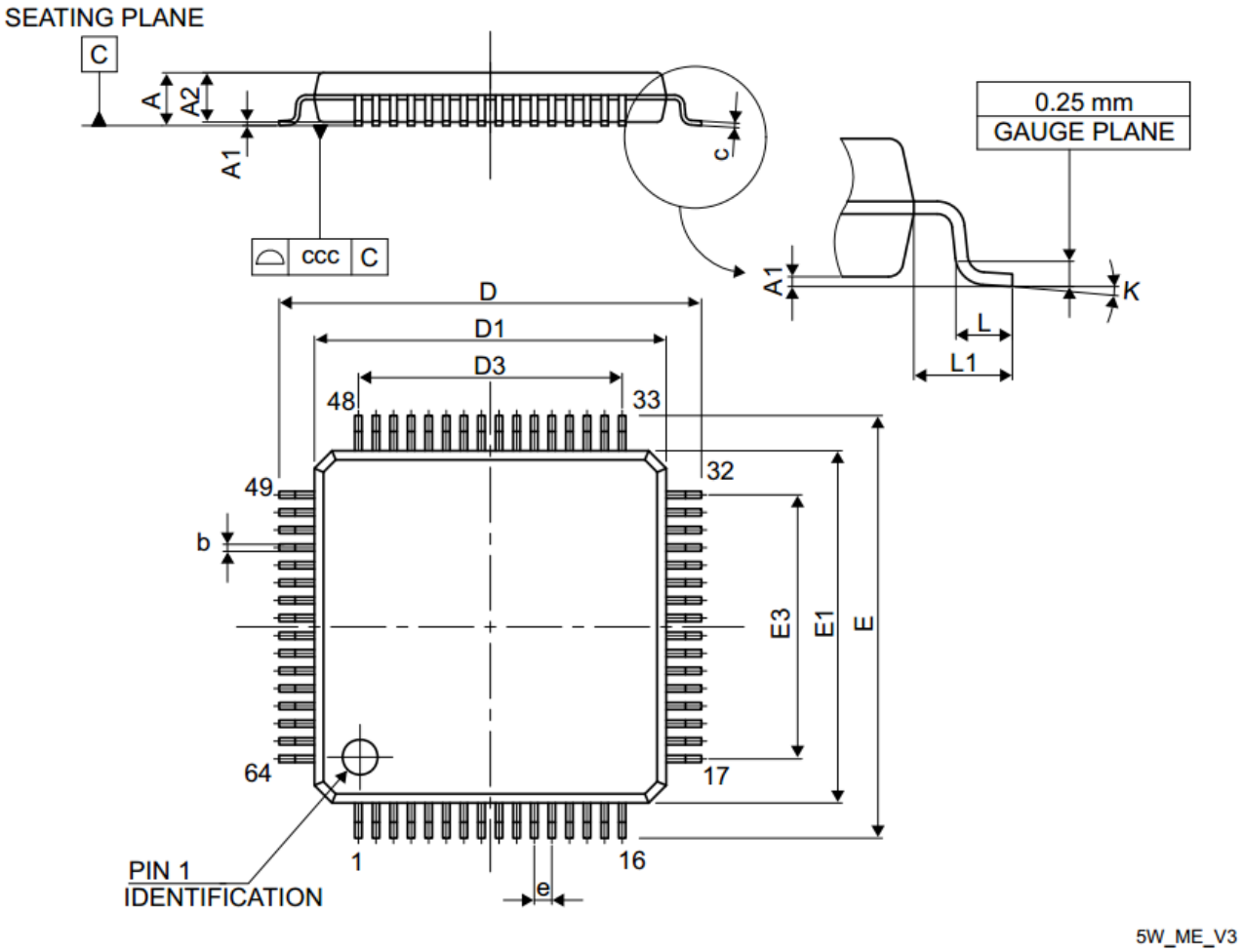
- Supporting the state-of-the-art E Ink panels
- SPI interface to host – slave device with additional /TC_EN and /TC_BUSY lines
- 2-bit color (4 grayscales)
- Complete solution including:
  - Temperature compensation
  - Common electrode voltage compensation
  - All voltages needed for the display
- Internal image buffer retains content during system power down
- Flashless 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

2 Outline

2.1 TC2

The information below applies to TC2-E97-320, TC2-E133-320 and TC2-E312-320.[1)]

STM32F446RET6: low-profile quad flat package outline; 64 terminals; body 10 x 10 x 1.40 mm



Unit		D	D_1	D_3	E	E_1	E_3	e	b	A	A_1	A_2	c	L	L_1	k[°]
mm	max	12.2	10.2	-	12.2	10.2	-	-	0.27	1.60	0.15	1.45	0.20	0.75	-	7.00
	nom	12.0	10.0	7.50	12.0	10.0	7.5	0.50	0.22	-	-	1.40	-	0.60	1.00	3.50
	min	11.8	9.80	-	11.8	9.80	-	-	0.17	-	0.05	1.35	0.09	0.45	-	0.00

Table 2.1: Dimensions (mm are the original dimensions)

2.2 TCM2-E97-320

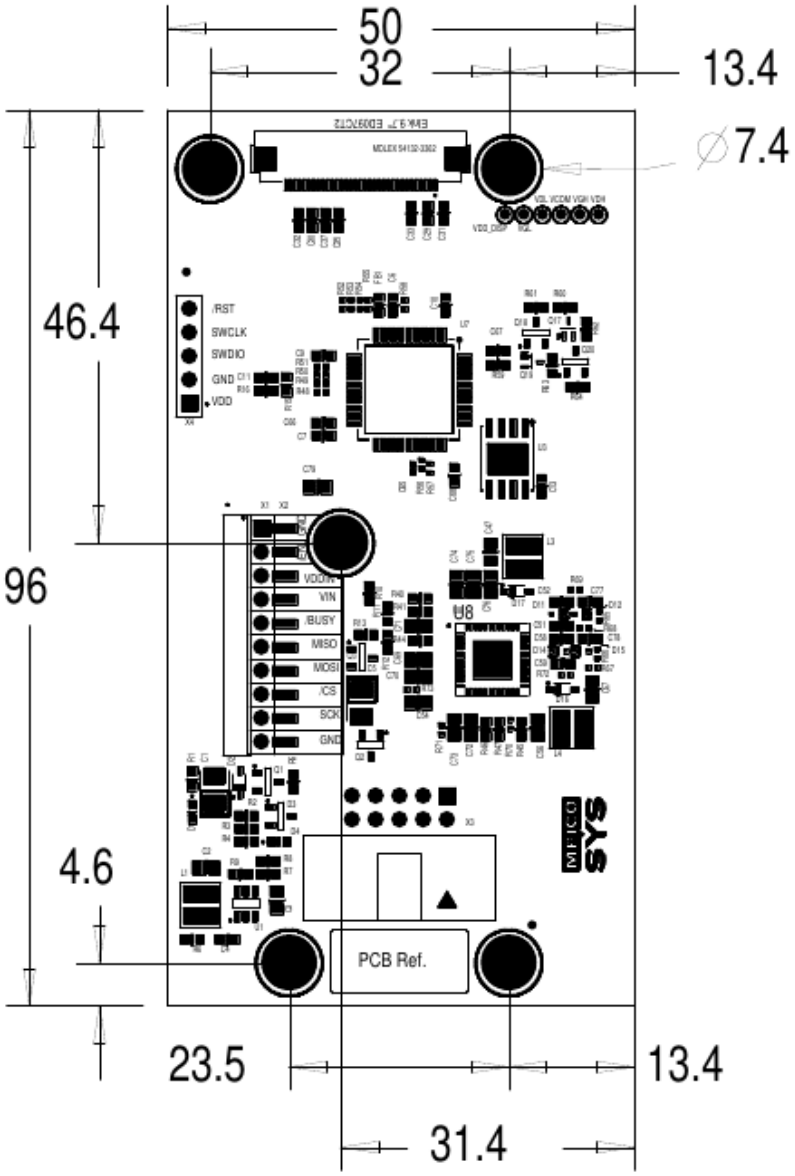


Figure 2.1: TCM2-E97-320 Outline (all dimensions in mm)

2.3 TCM2-E133-320

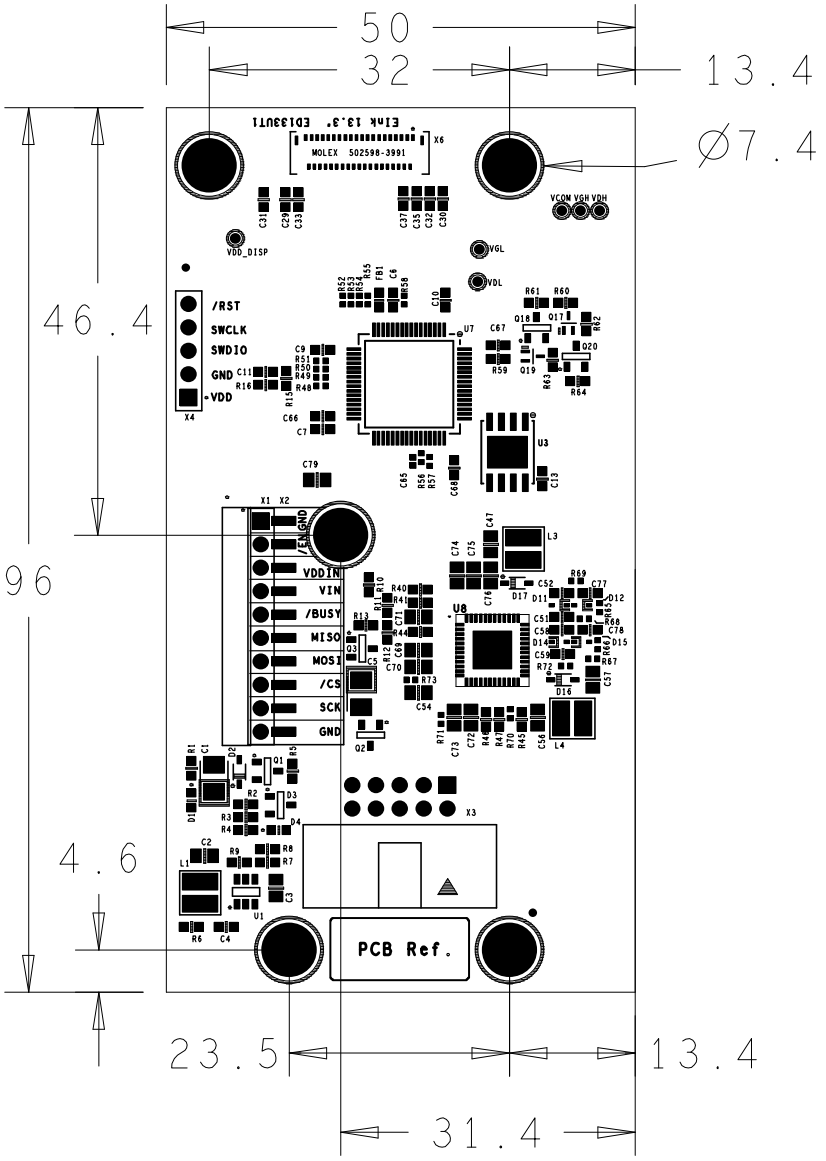


Figure 2.2: TCM2-E133-320 Outline (all dimensions in mm)

2.4 TCM2-E312-320

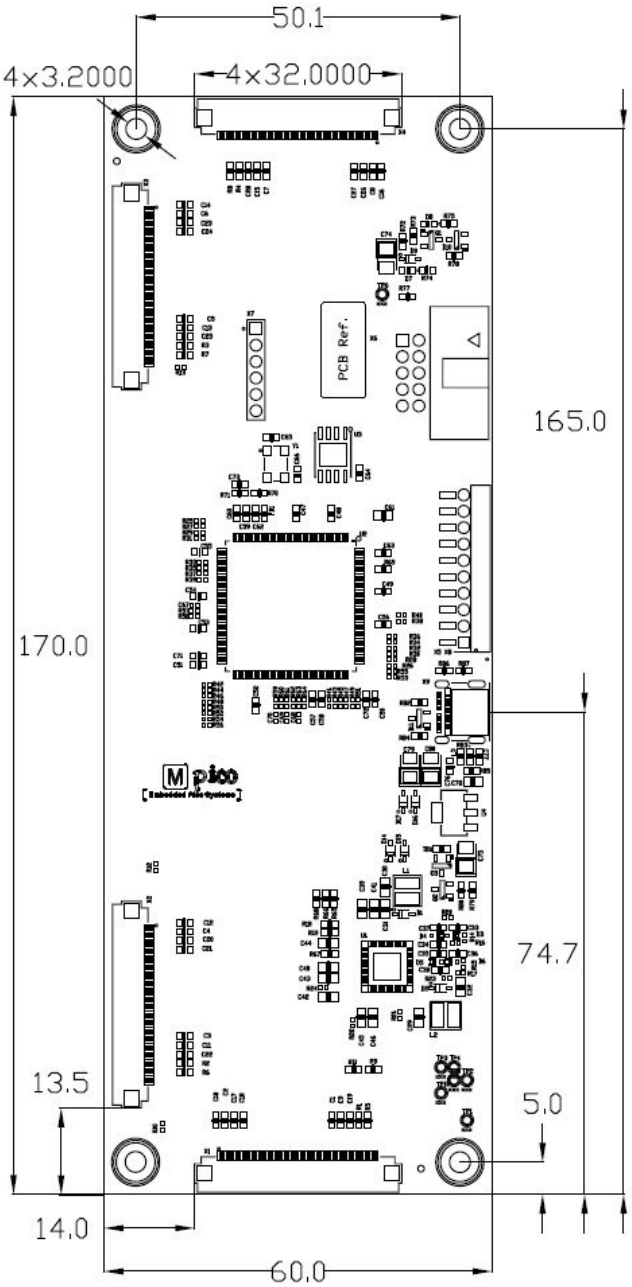


Figure 2.3: TCM2-E312-320 Outline (all dimensions in mm)

3 Characteristics

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

3.1 Operating Conditions

TCS2-E97-320, TCS2-E133-320 and TCS-E312-320

Symbol	Description	Min	Typ	Max	Unit
VDDIN	Standard digital operating voltage	2.7	3.0	3.6	V
VIN	Standard analog operating voltage	3.7	4.5	6	V
T_{op}	Operating temperature	0	+21	+50	°C

Table 3.1: Typical operating conditions – TCS2-E97-320, TCS2-E133-320 and TCS2-E312-320

3.2 Absolute Maximum Ratings

NOTE TCM2 features solder pads for overvoltage protection 3.6 V Zener diode (D2 in TCM2-E97-320, TCM2-E133-320 and D9 in TCM2-E312-320). 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-E97-320, TCS2-E133-320 and TCS2-E312-320

Symbol	Description	Min	Typ	Max	Unit
VDDIN	Digital supply voltage	–0.5	–	4.0	V
VIN	Analog supply voltage	–0.5	–	7.0	V
V_I	Logic input voltage	–0.5	–	4.6	V
T_{st}	Storage temperature	–20	–	+60	°C

Table 3.2: Absolute maximum ratings for TCS2-E97-320, TCS2-E133-320 and TCS2-E312-320

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;

Power supply:

- VDDIN = 3.3 V
- VIN[1)] = 3.7 V
- VIN[2)] = 5 V

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

VIN and VDDIN were measured separately, complete current or energy value is sum of VIN and VDDIN.

TCS2-E97-320

Symbol	Description	Operation	Typical	Max	Unit
IDD	Average digital part current consumption	Display update – Quality BWB	49.4	–	mA
		Display update – Quality WBW	41.9	–	mA
		Display update – Flashless	34.3	–	mA
		Display update – Flashless-Inverted	44.2	–	mA
		Data reception on SPI	16.7	–	mA
		Disabled (/TN_EN inactive)	<1	<1	µA
IIN	Average analog part current consumption	Display update – Quality BWB	23	68.4	mA
		Display update – Quality WBW	22.7	69.3	mA
		Display update – Flashless	16.4	135.8	mA
		Display update – Flashless-Inverted	24.4	192.7	mA
		Disabled (/TN_EN inactive)	<1	<1	µA
E	Average energy consumption in room temperature (total)	Display update – Quality BWB	515	860	mJ
		Display update – Quality WBW	705	817	mJ
		Display update – Flashless	113	401	mJ
		Display update – Flashless-Inverted	212	773	mJ
		2-bit image upload	35	–	mJ

Table 3.3: Supply current characteristics – TCS2-E97-320

1)TCM2-E133-320
2)TCM2-E312-320

TCS2-E133-320

Symbol	Description	Operation	Typical	Max	Unit
IDD	Average digital part current consumption	Display update – Quality BWB	44.1	–	mA
		Display update – Quality WBW	43.3	–	mA
		Display update – Flashless	37.3	–	mA
		Display update – Flashless-Inverted	47.5	–	mA
		Data reception on SPI	19.2	–	mA
		Disabled (/TN_EN inactive)	<1	<1	µA
IIN	Average analog part current consumption	Display update – Quality BWB	31.6	96.5	mA
		Display update – Quality WBW	29.5	97	mA
		Display update – Flashless	23	237.7	mA
		Display update – Flashless-Inverted	61.8	345.5	mA
		Disabled (/TN_EN inactive)	<1	<1	µA
E	Average energy consumption in room temperature (total)	Display update – Quality BWB	451	864	mJ
		Display update – Quality WBW	429	853	mJ
		Display update – Flashless	146	707	mJ
		Display update – Flashless-Inverted	381	1421	mJ
		2-bit image upload	61	–	mJ

Table 3.4: Supply current characteristics – TCS2-E133-320

TCS2-E312-320

Symbol	Description	Operation	Typical	Max	Unit
IDD	Average current consumption	Display update – Quality BWB	49.9	–	mA
		Display update – Quality WBW	49.4	–	mA
		Display update – Flashless	44.1	–	mA
		Display update – Flashless-Inverted	52.9	–	mA
		Data reception on SPI	20.3	–	mA
		Disabled (/TN_EN inactive)	<1	<1	µA
IIN	Average analog part current consumption	Display update – Quality BWB	35.3	286.9	mA
		Display update – Quality WBW	34.4	295.3	mA
		Display update – Flashless	27.5	512	mA
		Display update – Flashless-Inverted	43.6	674.3	mA
		Disabled (/TN_EN inactive)	<1	<1	µA
E	Average energy consumption in room temperature	Display update – Quality BWB	1020	4719	µA
		Display update – Quality WBW	975	4771	mJ
		Display update – Flashless	243	2327	mJ
		Display update – Flashless-Inverted	506	4574	mJ
		2-bit image upload	761	–	mJ

Table 3.5: Supply current characteristics – TCS2-E312-320

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 (VIN) [V]	Image Used for Measurement	Ambient Temp. [°C]
Average current consumption	Display update	Min	3.7	Transition *ESL* to *ESL*	21
		Max	3.7	Transition *Checkerboard* to *Checkerboard*	21
	Data reception on SPI	Min	3.7	*ESL*	21
		Max	3.7	*Checkerboard*	21
Average energy consumption in room temperature	Display update	Min	3.7	ESL	21
		Max	3.7	*Checkerboard*	21

Table 3.6: VIN Measurement results conditions

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

Table 3.7: VDD Measurement results conditions

Figure 3.1 ESL image for TCS2-E97, TCS2-E133 and TCS-E312

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 3.8: Typical operating conditions

4 Image Upload and Display Refresh Time

4.1 Image Upload Time

The image upload time was measured with a 2-bit uncompressed image. Uploading a 1-bit image will take half the time. The upload time can be reduced with the use of compression – proportionally to the compression rate.

TCM	Time [s]
TCM2-E97-320	0.41
TCM2-E133-320	0.8
TCM2-E312-320	6.5

Table 4.1: Image upload time

4.2 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.

4.3 BWB Transition

T_{amb} [°C]		0	5	10	15	20	25	30	35	40	45	50
Time [s]	TCS2-E97-320	4.96	3.60	2.66	2.39	2.11	1.98	1.98	1.71	1.57	1.44	1.44
	TCS2-E133-320	3.90	3.46	2.80	2.59	1.92	1.71	1.71	1.70	1.92	1.92	2.14
	TCS2-E312-320	6.73	5.16	4.54	4.22	3.28	2.65	2.65	2.96	2.96	3.28	3.59

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

4.4 WBW Transition

T_{amb} [°C]		0	5	10	15	20	25	30	35	40	45	50
Time [s]	**TCS2-E97-320**	4.96	3.60	2.66	2.39	2.11	1.98	1.98	1.71	1.57	1.44	1.44
	TCS2-E133-320	3.85	3.42	2.77	2.55	1.94	1.69	1.69	1.69	1.90	1.90	2.11
	TCS2-E312-320	6.54	5.02	4.41	4.10	3.19	2.58	2.58	2.88	2.89	3.19	3.49

Table 4.3: Display refresh time versus ambient temperature – WBW transition

4.5 Flashless Transition

T_{amb} [°C]		0	5	10	15	20	25	30	35	40	45	50
Time [s]	**TCS2-E97-320**	1.23	0.99	0.88	0.76	0.68	0.65	0.65	0.59	0.59	0.59	0.59
	TCS2-E133-320	1.03	0.92	0.82	0.82	0.72	0.72	0.72	0.76	0.76	0.76	0.81
	TCS2-E312-320	1.49	1.30	1.10	1.02	0.87	0.87	0.87	0.87	0.87	0.95	1.10

Table 4.4: Display refresh time versus ambient temperature – Flashless transition

4.6 Flashless-Inverted Transition

T_{amb} [°C]		0	5	10	15	20	25	30	35	40	45	50
Time [s]	**TCS2-E97-320**	2.05	1.59	1.36	1.08	0.96	0.9	0.9	0.79	0.78	0.78	0.78
	TCS2-E133-320	1.59	1.39	1.19	1.2	1.0	1.0	1.0	1.0	1.09	1.09	1.19
	TCS2-E312-320	2.54	2.13	1.72	1.57	1.29	1.29	1.29	1.29	1.29	1.43	1.71

Table 4.5: 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 TCM2 Interconnection

5.1.1 USB-C

TCS2-E312-320 is equipped with USB-C connector for power supply. It is operated in accordance with the USB Type-C™ standards.

5.1.2 Host Connector

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	Power supply for digital part
4	VIN	Power supply for analog part
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

5.2 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-E97-320, TCS2-E133-320 and TCS-E312-320, VIN can either be supplied directly from the battery (e.g. coin-cell, li-po, li-ion) for improved efficiency, or can be shorted to VDDIN.

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

5.3 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

By the example of TCS2-E133-320:

The below timing diagram (Figure 5.1) represents the startup and initialization sequence after power-up. The TCS2-E133-320 is ready for communication after $T_{STARTUP}$ + T_{INIT} which is indicated by /TC_BUSY rising edge.

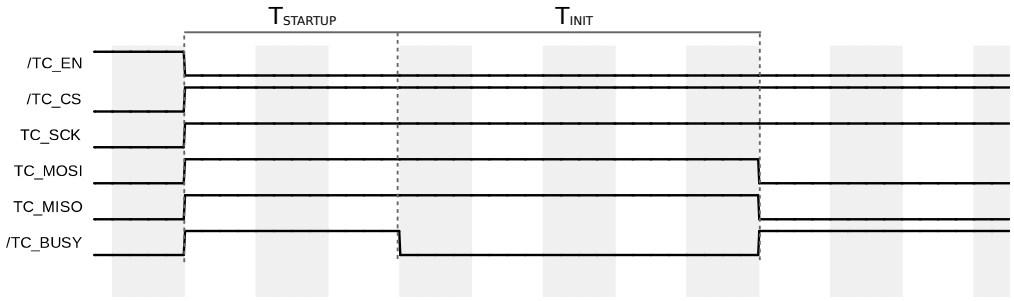


Figure 5.1: TCS2-E133-320 initialization sequence

1) Logical AND function is performed in case of many TC sharing the same BUSY line.

Time	Min	Max	Unit
$T_{STARTUP}$	-	12.5	ms
T_{INIT}	13.3	14.0	ms

Table 5.2: TCS2-E97-320, TCS2-E133-320 and TCS2-E312-320 startup and initialization times

SPI Settings

- Bit rate – up to 10.5 MHz
  - Effective bit rate: 6 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

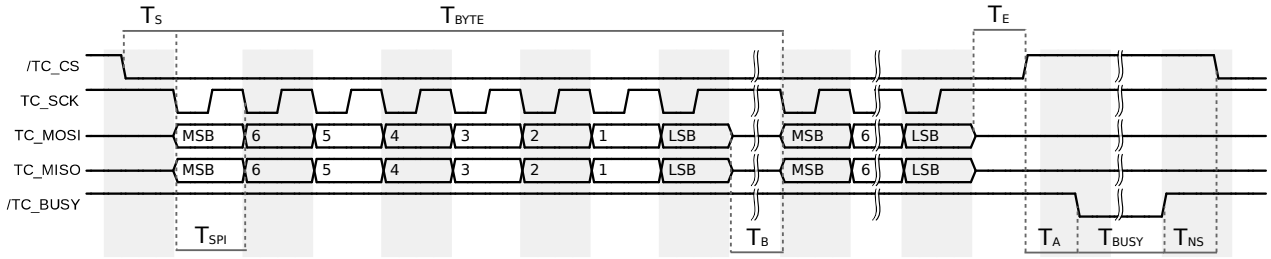


Figure 5.2: SPI timing diagram

Reference SPI timing diagram below:

Time	T_S	T_{BYTE}	T_{SPI}	T_B	T_E	T_A	T_{BUSY}	T_{NS}
Min.	50 ns	855 ns	95 ns[1]	94 ns[2]	50 ns	–	10 µs	0
Max.	–	–	1 ms	–	–	3.8 µs	–	–

Table 5.3: TCS-97-320_v1.0, TCS2-E133-320_v1.3 and TCS2-E312-320_v1.2 SPI timing description.

Communication Flow

TCS2 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.

1) Minimum T_{SPI} value reflects the maximum supported bit rate of 10.5 MHz. In this case T_B has to be greater than or equal to 94 ns.
2) T_{SPI} value can equal 0 if frequency is lower than or equal to 6 MHz.

5.4 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 15(for E97), 7 (for E133) and 3 (for E312). The default 32 Mbit memory can be replaced with a more sizable memory chip, allowing greater slots number.

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

5.4.1 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.

5.5 Use Cases

5.5.1 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 5.6.1.1 UploadImageData command with argument P2=0 to fill one of the framebuffer slots with image data
2) Evoke 5.6.2.1 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

5.5.2 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 5.6.2.1 DisplayUpdate. Generated image has to be without header in type0 (without first 16 bytes .epd file).

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.

For 2-bit Partial Image Upload user has to perform several steps which contain:

- Erasing slot by command 5.6.1.5 ImageEraseFrameBuffer
- Loading 2bit Header by 5.6.1.1 UploadImageData command
- Setting given ROI by 5.6.1.6 ImageUploadSetROI command
- Upload image data (alternately loading the data lines of the most bits and the data lines of the least bits – described in 6.1.2 Pixel Data Format Type 0 – 2-bit) by 5.6.1.1 UploadImageData command

Example use case:

1) Erase chosen slot (see 2-byte status code) – erase is recommended before any partial upload operation
2) Evoke 5.6.1.6 ImageUploadSetROI command to define the area (e.g. 100x200 px in the upper-right corner) that is about to be modified
3) Evoke 5.6.1.1 UploadImageData commands to fill the above-defined area with image data
4) Evoke 5.6.1.6 ImageUploadSetROI command to define the next area (areas cannot intersect)
5) Use 5.6.1.8 ImageUploadCopySlots command to copy image data from another slot to the defined area
6) Use 5.6.1.6 ImageUploadSetROI command to define the next area (areas cannot intersect)
7) Use 5.6.1.7 ImageUploadFixVal to e.g. fill the defined area with white color
8) Evoke 5.6.2.1 DisplayUpdate command to display created image
9) The sequence of commands (2-3, 4-5, 6-7) can be run in any order

5.6 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
0x9EXX	EP_DCDC_ERROR	Power management IC reports error

Figure 5.3: 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

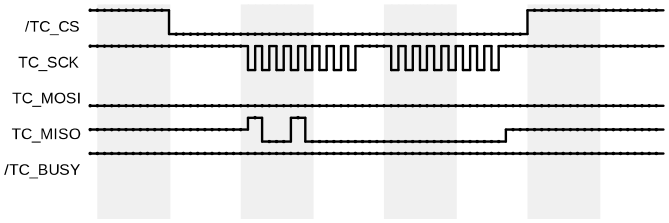


Figure 5.4: Example readout - 0x9000 response

5.6.1 Image Data Commands

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

5.6.1.1 UploadImageData

Command

INS	P1	P2	Lc	Data
0x20	0x01	Si	Data packet size (max 0xFB)	[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 0 in TCS2-97-320_v1.0
- Type 0 in TCS2-E133-320_v1.3
- Type 7 in TCS2-E312-320_v1.2

Parameters

- P2: Si – framebuffer slot number (see 5.4 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

5.6.1.2 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 5.4 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

5.6.1.3 GetChecksum

Command

INS	P1	P2	Le
0x2E	0x01	Si	0x02

Description

Get 16-bit checksum of an image stored in the TCS2 memory.

NOTE Checksum is calculated based on Pixel Data Format Type used to store images.

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 5.4 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

5.6.1.4 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

5.6.1.5 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 5.4 Framebuffer Slots)

Response

- 2-byte status code

5.6.1.6 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: Si – framebuffer slot number (see 5.4 Framebuffer Slots)

Data

ROI_data: four 16-bit (MSB first) values that define ROI area:

- Xmin (inclusive), Xmax (exclusive) (from 0 to display’s maximum X resolution),
- Ymin (inclusive), Ymax (exclusive) (from 0 to display’s maximum Y resolution).

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

5.6.1.7 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 5.4 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

5.6.1.8 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 5.4 Framebuffer Slots)

Data

- Si_Source – source framebuffer slot number (see 5.4 Framebuffer Slots)

Response

- 2-byte status code

5.6.2 Display Control Commands

5.6.2.1 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	BWB (Default)	Default transition sequence – with black-white-black screen flashing. Offers the best image quality.
0x82	WBW	Transition sequence – with white-black-white 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. Flashless update can be used also with 2-bit images but it doesn't guarantee best possible quality. To achieve best image quality always display a 1-bit image with a Quality (WBW or BWB) drivescheme before using FLS or FLS_INV drivescheme.
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 5.4 Framebuffer Slots)

Data

- Temp (optional) – one byte defining temperature in degrees Celsius, U2 encoded, used for temperature compensation substituting the actual temperature readings
- Response
- 2-byte status code

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

5.6.2.2 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-E97-320_v1.0” terminated by 0x00 byte] + 0x9000 status code in case of TC2-E97-320_v1.0
- [String: “MpicoSys TC2-E133-320_v1.3” terminated by 0x00 byte] + 0x9000 status code in case of TC2-E133-320_v1.3
- [String “MpicoSys TC2-E312-320_v1.2” terminated by 0x00 byte] + 0x9000 status code in case of TC2-E312-320_v1.2
- 2-byte error status code

5.6.2.3 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

5.6.3 System Info Commands

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

5.6.3.1 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-E97-320_fA_BIN” terminated by 0x00 byte] + 0x9000 status code in case of TC2-E97-320_v1.0
- [String: “MpicoSys TC2-E133-320_fD_BIN” terminated by 0x00 byte] + 0x9000 status code in case of TC2-E133-320_v1.3
- [String: “MpicoSys TC2-E312-320_fC_BIN” terminated by 0x00 byte] + 0x9000 status code in case of TC2-E312-320_v1.2
- 2-byte error status code

5.6.3.2 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 B4 00 00 00 00 00 00 43 0F 05 00 00 00 00 00 + 0x9000 status code in case of TCS2-E97-320_v1.0
- 0x D0 B2 00 00 00 00 00 00 3E 07 05 00 00 00 00 00 + 0x9000 status code in case of TCS2-E133-320_v1.3
- 0x D0 B3 00 00 00 00 00 00 3F 03 05 00 00 00 00 00 + 0x9000 status code in case of TCS2-E312-320_v1.2
- 2-byte error status code

5.6.4 Sensor Data Commands

5.6.4.1 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

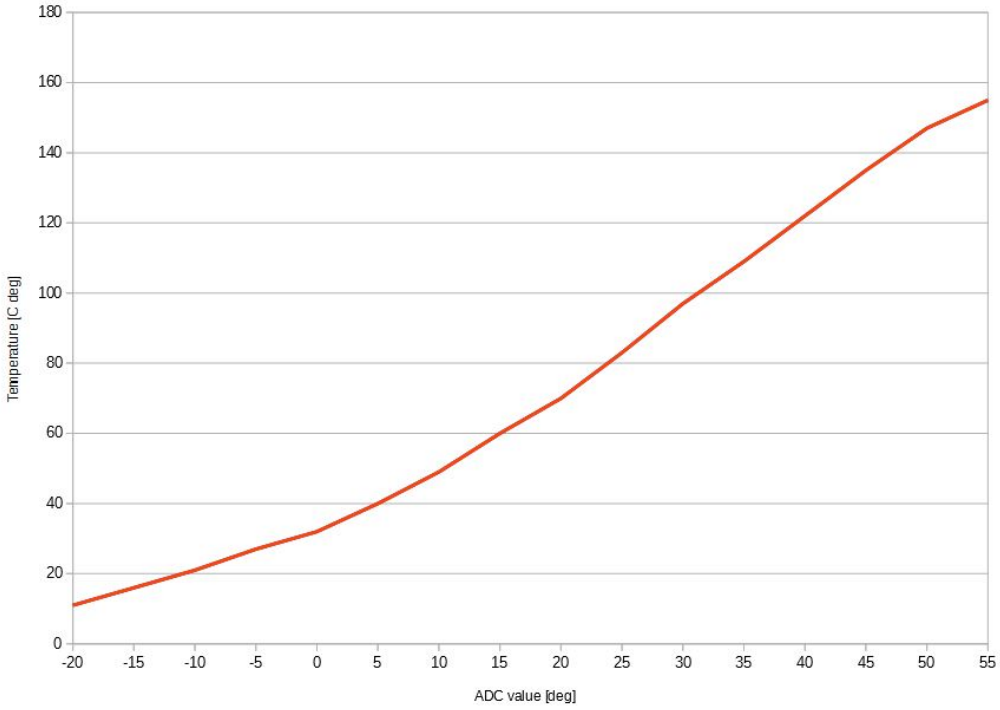


Figure 5.5: Ambient temperature versus ADC readout chart

Temperature [°C]	ADC value [dec]	ADC value [hex]
-20	11	B
-15	16	10
-10	21	15
-5	27	1B
0	32	20
5	40	28
10	49	31
15	60	3C

Temperature [°C]	ADC value [dec]	ADC value [hex]
20	70	46
25	83	53
30	97	61
35	109	6D
40	122	7A
45	135	87
50	147	93
55	155	99

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]
9.7”	1200x825	1	16	123,600	123,616
		2	16	247,200	247,216
13.3″	1600×1200	1	16	240,000	240,016
		2	16	480,000	480,016
31.2″	1440×2560	1	16	460,800	460,816
		2	16	921,600	921,616

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	0x43	Panel code	9.7”
		0x3E		13.3″
		0x3F		31.2″
X res	2 bytes	0x04B0	1200 px	
		0x05A0	1440 px	
		0x0640	1600 px	
Y res	2 bytes	0x0339	825 px	
		0x04B0	1200 px	
		0x0A00	2560 px	
Color depth	1 byte	0x01	Image color depth	1-bit (black and white)
		0x02		2-bit (4 grayscales)
Pixel Data Format Type	1 byte	0x00	Image pixel data format type 0	
		0x07	Image pixel data format type 7	
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-E97-320 1-bit: 0x 43 04B0 0339 01 00 00000000000000000
- TCS2-E97-320 2-bit: 0x 43 04B0 0339 02 00 00000000000000000
- TCS2-E133-320 1-bit: 0x 3E 0640 04B0 01 00 00000000000000000
- TCS2-E133-320 2-bit: 0x 3E 0640 04B0 02 00 00000000000000000
- TCS2-E312-320 1-bit: 0x 3F 05A0 0A00 01 07 00000000000000000
- TCS2-E312-320 2-bit: 0x 3F 05A0 0A00 01 07 00000000000000000

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:

- 0b0 – white
- 0b1 – black

2-bit gray scale provides 4 colors:

- 0b00 – white
- 0b01 – light-gray
- 0b10 – dark-gray
- 0b11 – black

6.1.1 Pixel Data Format Type 0 – 1-bit

This format is used in TCS2-E97-320 and TCS2-E133-320. 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 1-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;
    }
```

6.1.2 Pixel Data Format Type 0 – 2-bit

This format is used in TCS2-E97-320 and TCS2-E133-320.

The EPD file in this type is formed from two byte sequences per each consecutive row of the original image data – a sequence of bytes containing most significant color bits for each pixel, followed by a sequence of bytes containing least significant color bits for each pixel. The length of each sequence is determined by the ePaper panel width divided by 8 (200 bytes for TCM2-E133-320).

This is described in the table below.

Image row No.	Field	Size [bytes]	Description
–	EPD image header	16	
1	MSb image data line	[Width]/8	Sequence of bytes containing **most** significant color bits for each pixel
1	LSb image data line	[Width]/8	Sequence of bytes containing **least** significant color bits for each pixel
2	MSb image data line	[Width]/8	Sequence of bytes containing **most** significant color bits for each pixel
2	LSb image data line	[Width]/8	Sequence of bytes containing **least** significant color bits for each pixel
..	MSb image data line	[Width]/8	Sequence of bytes containing **most** significant color bits for each pixel
..	LSb image data line	[Width]/8	Sequence of bytes containing **least** significant color bits for each pixel
[Height]	MSb image data line	[Width]/8	Sequence of bytes containing **most** significant color bits for each pixel
[Height]	LSb image data line	[Width]/8	Sequence of bytes containing **least** significant color bits for each pixel

Table 6.5: Pixel Data Format Type 0 – 2-bit description

Conversion Algorithm

The algorithm for conversion from standard RAW 2-bit data to EPD type 0 format is described below, on an example of an image of 16-pixel width.

- Start with a byte array of image data which is already downsampled to 2-bit palette; each byte conveys information on 1 pixel

1) Get a single row of 16 bytes (16 pixels):

Input byte No.:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Byte value (binary):	00	10	11	11	10	01	11	01	11	10	10	00	01	00	11	01

Table 6.6: Input data

2) Merge each eight input byte MSb values (numbered from 0 to 7 and from 8 to 15) into one output byte, conveying information on 8 pixels. Do the same for LSb.

Input byte No.:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pixel value (binary):	00	10	11	11	10	01	11	01	11	10	10	00	01	00	11	01
Output bytes value (1st sequence)	0b01111010 (0x7A)								0b11100010 (0xE2)							
Output bytes value (2nd sequence)	0b00110111 (0x37)								0b10001011 (0x8B)							

Table 6.7: Output data – single byte

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

6.1.3 Pixel Data Format Type 7 – 1-bit

This format is used in TCS2-E312-320.

Conversion Algorithm

The algorithm for conversion from standard RAW 2-bit data to EPD format is described below. The algorithm is described on a 16x8 pixel image, but can be extrapolated to the actual panel resolution.

- Start with a byte array of pixel data which is already downsampled to 1-bit grayscale. The value of a single byte can be 0x0 or 0x1
- Get a single row of sixteen pixels:

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pixel No.:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pixel Value	0	1	1	1	1	0	1	0	1	1	1	0	0	0	1	1

- Get a set of 16 pixels (16 bytes) and do the following (numbering from 0 to 15):
  - Byte 0: move to position 8
  - Byte 1: move to position 10
  - Byte 2: move to position 12
  - Byte 3: move to position 14
  - Byte 4: move to position 0
  - Byte 5: move to position 2
  - Byte 6: move to position 4
  - Byte 7: move to position 6
  - Byte 8: move to position 9
  - Byte 9: move to position 11
  - Byte 10: move to position 13
  - Byte 11: move to position 15
  - Byte 12: move to position 1
  - Byte 13: move to position 3
  - Byte 14: move to position 5
  - Byte 15: move to position 7
- Get the next set of 8 pixels and repeat the sequence.
- Repeat until all the pixels are processed:

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pixel No.:	4	12	5	13	6	14	7	15	0	8	1	9	2	10	3	11
Pixel Value	1	0	0	0	1	1	0	1	0	1	1	1	1	1	1	0

- Merge 16 pixels (2bytes of pixel data) into one byte conveying information on 8 pixels:

Bytes No.:	0	1
Byte Value:	0x8D | 0b10001101	0x7E | 0b01111110

Sample Code

Below is sample Python code for image conversion:

```
def __convert(self, img):
        (img_width, img_height) = img.size
        assert img_width == 1440
        assert img_height == 640*4

        img=img.convert('P')
        img=img.transpose(Image.FLIP_LEFT_RIGHT)
        imgbytes = list(img.tobytes())
        imgnewbytes=list(len(imgbytes)*b'\x00')
```

```
        trans1=( # new, original
            (6,7),
            (4,6),
            (2,5),
            (0,4),
            (14,3),
            (12,2),
            (10,1),#
            (8,0),
            (7,15),
            (5,14),
            (3,13),
            (1,12),
            (15,11),
            (13,10),
            (11,9),##
            (9,8)
           )

        for i in range(len(imgbytes)/16):
            for t,f in trans1:
                imgnewbytes[16*(i)+t]=imgbytes[16*(i)+f]

        imgnew = Image.frombytes('P',img.size,''.join(imgnewbytes))
        palette = []
        for i in range(256):
            p=0xFF
            if i>0:
                p=0x00
            palette.extend((p, p, p))
        assert len(palette) == 768
        imgnew.putpalette(palette)
        newstr=imgnew.convert('1').tobytes()

        return newstr
```

6.1.4 Pixel Data Format Type 7 – 2-bit

This format is used in TCS2-E312-320.

The EPD file in this type is formed from two byte sequences per each consecutive row of the original image data – a sequence of bytes containing most significant color bits for each pixel, followed by a sequence of bytes containing least significant color bits for each pixel. The length of each sequence is determined by the ePaper panel width divided by 8 (200 bytes for TCM2-E312-320).

This is described in the table below.

Image row No.	Field	Size [bytes]	Description
–	EPD image header	16	
1	MSb image data line	[Width]/8	Sequence of bytes containing **most** significant color bits for each pixel
1	LSb image data line	[Width]/8	Sequence of bytes containing **least** significant color bits for each pixel
2	MSb image data line	[Width]/8	Sequence of bytes containing **most** significant color bits for each pixel
2	LSb image data line	[Width]/8	Sequence of bytes containing **least** significant color bits for each pixel
..	MSb image data line	[Width]/8	Sequence of bytes containing **most** significant color bits for each pixel
..	LSb image data	[Width]/8	Sequence of bytes containing **least** significant color bits for

	line		each pixel
[Height]	MSb image data line	[Width]/8	Sequence of bytes containing **most** significant color bits for each pixel
[Height]	LSb image data line	[Width]/8	Sequence of bytes containing **least** significant color bits for each pixel

Table 6.8: Pixel Data Format Type 7 – 2-bit description

Conversion Algorithm

The algorithm for conversion from standard RAW 2-bit data to EPD type 0 format is described below, on an example of an image of 16-pixel width.

- Start with a byte array of image data which is already downsampled to 2-bit palette; each byte conveys information on 1 pixel

1) Get a single row of 16 bytes (16 pixels):

Input byte No.:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Byte value (binary):	00	10	11	11	10	01	11	01	11	10	10	00	01	00	11	01

Table 6.9: Input data

2) Merge each eight input byte MSb values (numbered from 0 to 7 and from 8 to 15) into one output byte, conveying information on 8 pixels. Do the same for LSb.

Input byte No.:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pixel value (binary):	10	01	01	00	11	11	01	01	00	11	10	10	11	10	11	00
Output bytes value (1st sequence)	0b10001100 (0x8C)								0b01111110 (0x7E)							
Output bytes value (2nd sequence)	0b01101111(0x6F)								0b01001010(0x4A)							

Table 6.10: Output data – single byte

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

7 Revision History

Document Revision	Release Date	Document Status	Supersedes
E	2018-10-05	Approved	D
D	2018-08-06	Draft	C
C	2018-07-13	Draft	B
B	2018-07-06	Draft	A
A	2017-07-01	Approved	–

Table 7.1: Revision history

Document Revision	Change Log
E	Added TCS2-E97-320_v1.0, revision for TCS2-E312-320_v1.2, updated TCM2 Supply Current Characteristics and Type 7 description
D	Revision for TCS2-E133-320_v1.3 and TCS2-E312-320_v1.1
C	Removed SetSlotsNumber subsection
B	Revision for TCS2-E133-320_v1.2 and TCS2-E312-320_v1.1
A	Initial version

Table 7.2: Change log

8 Legal Information

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.