Date sheet (Appliction note) #710

Tango C32/C44/C48

October 19, 2011

Abstract

This document describes Tango C32/C44/C48 usage.

Contents

Revision history

Version	Content	Pages	Date	Author	Reviewed
0	Initial release	59	23rd of May 2011	Jeff Apollo	LPO
1	Add figure 2: C48 pins configuration. Add table 2: C48 pins description. Add figure 1: C32 pins configuration. Add table 1: C32 pins description. Add figure 25: C32 circuit Add figure 25: C44 circuit Add figure 26: C48 circuit Add figure 27: C44+C32 circuit And other information about C48 Add the figure 29 figure 31 show in sleep and active mode Add the figure 78 for the C32 package Add figure 67: for EEP_M0_X_THR_ONEFINGER parameter Redraw the figure 73: in EEP_M0_X_THR_ONEFINGER section	67	26 of May 2011	Jeff Apollo	LPO
2	Modify the figure 6, figure 7 and figure 15	67	10 of june 2011	Jeff	LPO Marc
3	Modify the figure 24 figure 25 , figure 26 and figure 27 updata EEPROM content figure 75 figure 76 figure 74	70	22th June 2011	Jeff apollo	
4	Add the table 5 Add the table 6	70	20th July 2011	Jeff	Jeffery Alex Janson
5	Modify the figure 24 figure 25 figure 26 figure 27 Add the Principle of the buttons	78	29th July 2011	Jeff	nicole
6	Modify the Tango Core button circuits table 20 Add table 28 figure 79 figure 80 EEPROM parameter	78	08/25/11	Apollo	Marc Angela

Version	Content	Pages	Date	Author	Reviewed
7	For 2 fingers FW only Modify the figure 25 figure 25 figure 26 figure 61 figure 25 figure 26 figure 27 figure 61 add Sleep_freq and Auto_sleep_delay in table 14 Add section and figures for INT_Mode=11 Add eeprom parameters for button, INT mode, power mode, finger_sep... Add description for pinmap ruler and Raw Ctrl register	78	2011-10-19	Apollo	Jeffrey Massimo Masa Yannick

1 Electrical specification of Tango C

1.1 Tango C series

The Tango C series includes the Tango C32, Tango C44 and Tango C48. They are in common I/O pins and when not specified in this document, the description is valid for the C32, C44 and C48.

1.2 Pin description

Figure 1,figure 2 and figure 3 shows one chip solution (TANGO C32 40NQ)/(TANGO C44 56NQ)/(TANGO C48 56NQ) pins configuration. The table 1, table 2 and table 3are summarizes the pin functions of Tango C32/C44/C48.

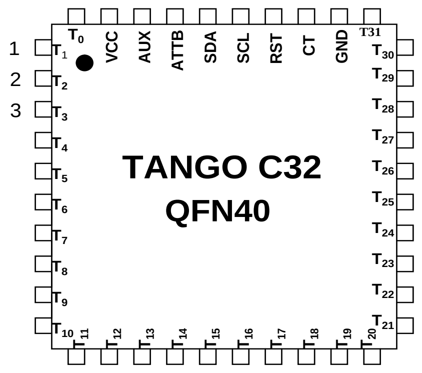


Figure 1: C32 pins configuration (top view)

Table 1: C32 pins description

VCC	Positive supply
GND	Negative supply
SDA	IIC data input
SCL	IIC clock input
ATTB	Bidirectional GPIO
AUX	Bidirectional GPIO
RST	Reset (active high)
CT	Optional tank capacitor
T_0..T_{31}	32 Tango Sensor lines

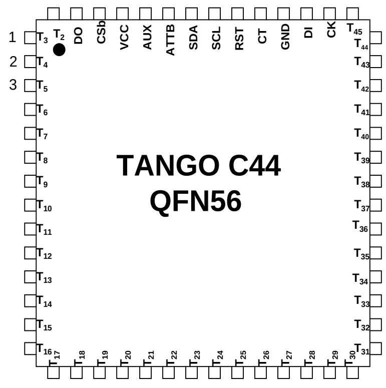


Figure 2: C44 pins configuration (top view)

Table 2: C44 pins description

VCC	Positive supply
GND	Negative supply
SDA	IIC data input
SCL	IIC clock input
ATTB	Bidirectional GPIO
AUX	Bidirectional GPIO
RST	Reset (active high)
CT	Optional tank capacitor
DO	Optional external Tango T2M
CK	Optional external Tango clock
Csb	Optional external Tango chip select
DI	Optional external Tango M2T
$T_2..T_{45}$	44 Tango Sensor lines

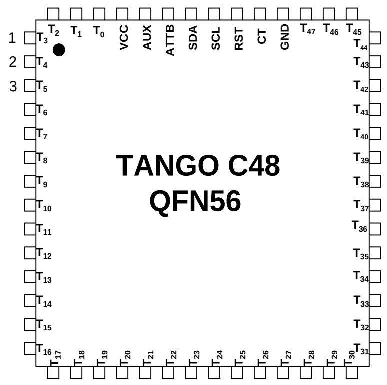

TANGO C48
QFN56
1
2
3
T_2 T_1 T_0 VCC AUX ATTB SDA SCL RST CT GND T_{47} T_{46} T_{45}
T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10} T_{11} T_{12} T_{13} T_{14} T_{15} T_{16}
T_{17} T_{18} T_{19} T_{20} T_{21} T_{22} T_{23} T_{24} T_{25} T_{26} T_{27} T_{28} T_{29} T_{30}
T_{31} T_{32} T_{33} T_{34} T_{35} T_{36} T_{37} T_{38} T_{39} T_{40} T_{41} T_{42} T_{43} T_{44}

Figure 3: C48 pins configuration (top view)

Table 3: C48 pins description

VCC	Positive supply
GND	Negative supply
SDA	IIC data input
SCL	IIC clock input
ATTB	Bidirectional GPIO
AUX	Bidirectional GPIO
RST	Reset (active high)
CT	Optional tank capacitor
$T_0..T_{47}$	48 Tango Sensor lines

1.2.1 VCC

Positive supply. It is recommended to decouple VCC to GND with 100nF, with shortest connection length.

1.2.2 GND

Negative supply.

1.2.3 RST

Figure 4: Reset input, active high, with low threshold voltage. A high level above V$_{RST,HI}$ for a duration longer that T$_{RST,IN}$ reset the MCU. When the MCU is in reset state, the four I/O SDA, SCL, ATTB and AUX are high impedance. Reset state, with RST pin high, is not a power saving mode. Reset is designed to be driven by the host processor, and maintained at low level during normal operation of the sensor. It is recommended to decouple RST pad to ground with a capacitor. Electrical characteristics of the pin are summarize in table 4 and illustrated in figure 5.

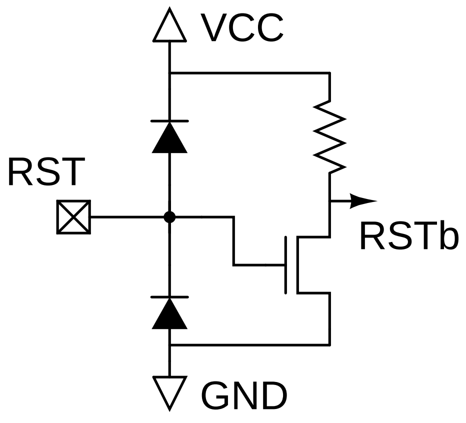


Figure 4: RST pin circuit

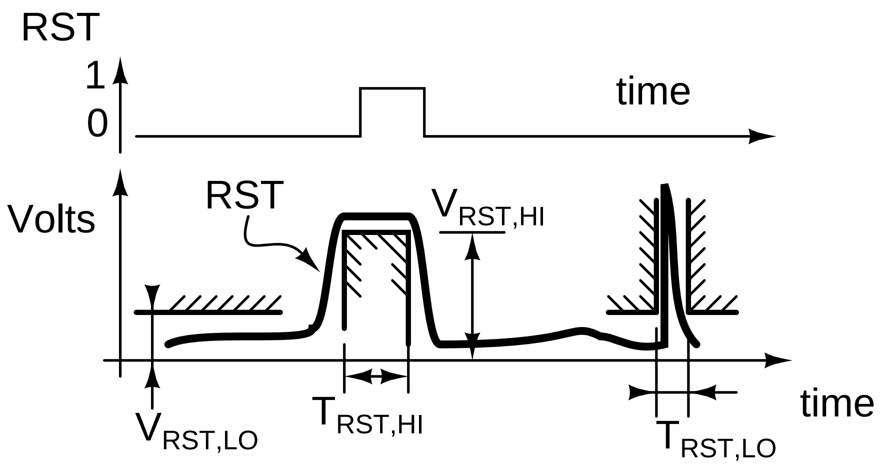


Figure 5: Reset timing

Table 4: RST pin characteristics

Symbol	Parameter	Min	Typ	Max	Unit
$V_{RST,HI}$	RST pad input level to activate a reset	0.9			V
$V_{RST,LO}$	RST pad input level to prevent a reset			0.4	V
$T_{RST,HI}$	Minimum duration of positive pulse applied on RST pad above $V_{RST,HI}$ to ensure a reset	80			ns
$T_{RST,LO}$	Maximum duration of positive pulse applied on RST pad above $V_{RST,HI}$ being rejected by glitch suppressor (no reset).			40	ns

1.2.4 SDA

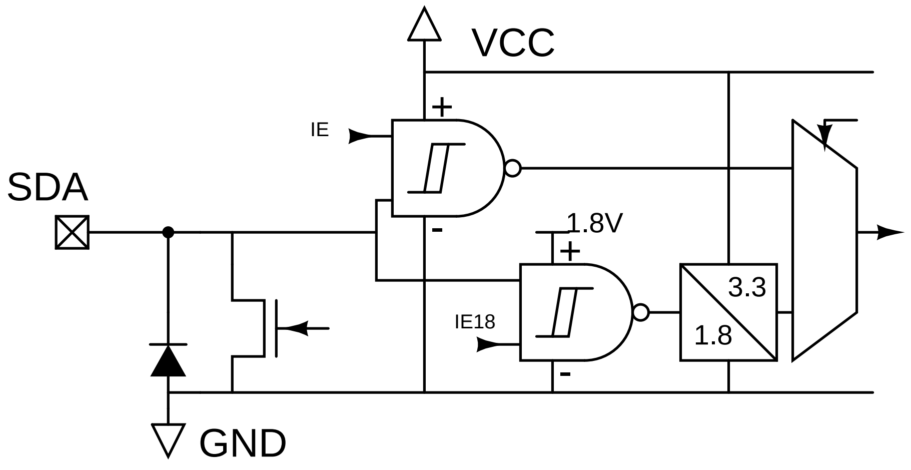


Figure 6: SDA pin circuit

Figure 6: IIC data. This pad input logic level can be selected to be near$0.5 \cdot V_{CC}$ or near 0.9 Volts pull up resistor is not provided by Tango C44/C48.

1.2.5 SCL

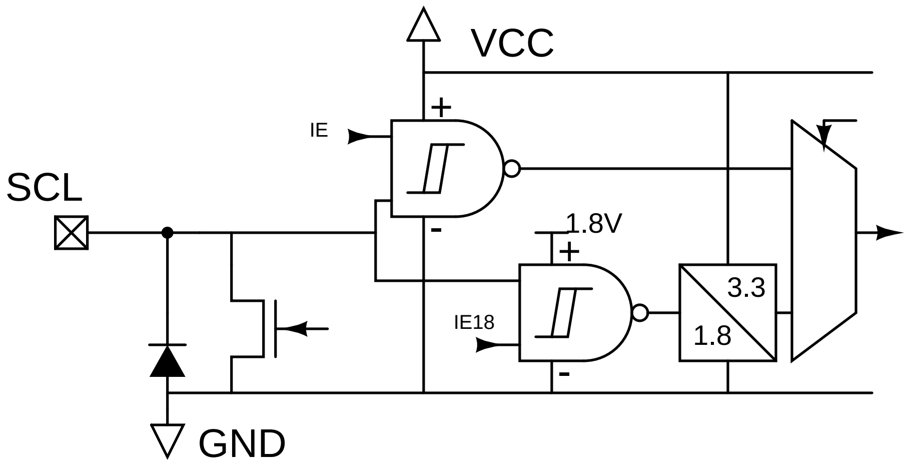


Figure 7: SCL pin circuit

Figure 7 : IIC clock. This pad input logic level can be selected to be near $0.5 \cdot V_{CC}$ or near 0.9 Volts. Pull up resistor is not provided by Tango C44/C48.

1.2.6 ATTB

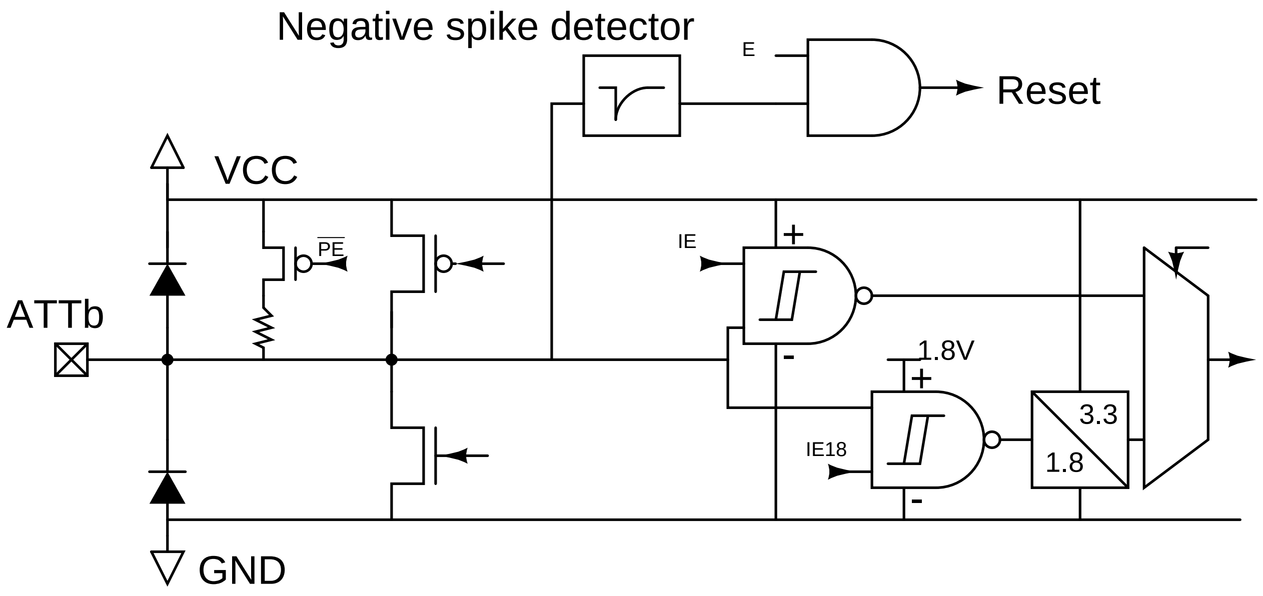


Figure 8: ATTB pin circuit

Figure 8: General purpose I/O. This pad input logic level can be selected to be near $0.5 \cdot V_{CC}$ or near 0.9 Volts. Pull up resistor can be enabled. Driving stage can be open drain or push-pull. ATTB is generally used by the firmware to attract the attention of the IIC master, for example to signal a touch. Optionally, ATTB can be configured as AC negative spike detector and used as a reset source.

1.2.7 CT

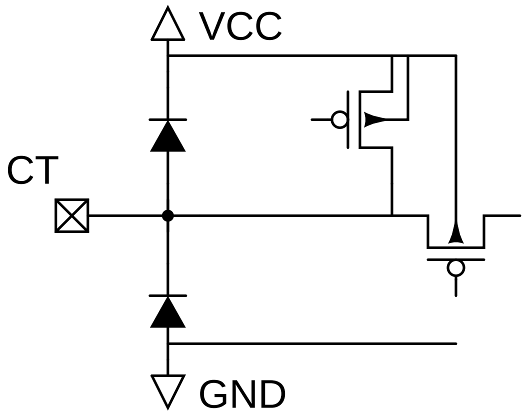


Figure 9: CT pin circuit

Figure 9: Tank capacitor. Optionally connect a tank capacitor to reduce effect of power supply noise (VCC to GND ripple) on the measurement.

1.2.8 CSb

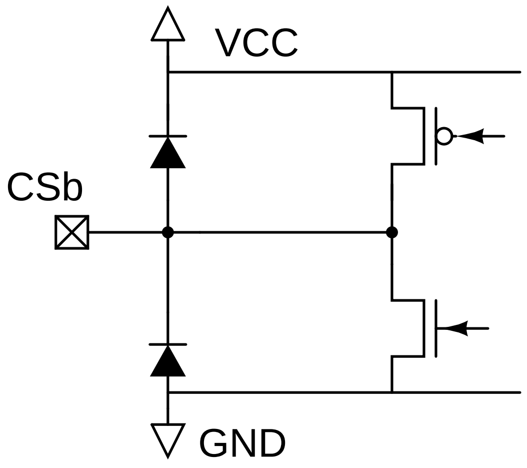


Figure 10: CSb pin circuit

CSb is only available with Tango C44. Figure 10: External Tango chip select. This output can drive a Tango F32 or F48 CSb input pad.

1.2.9 DI

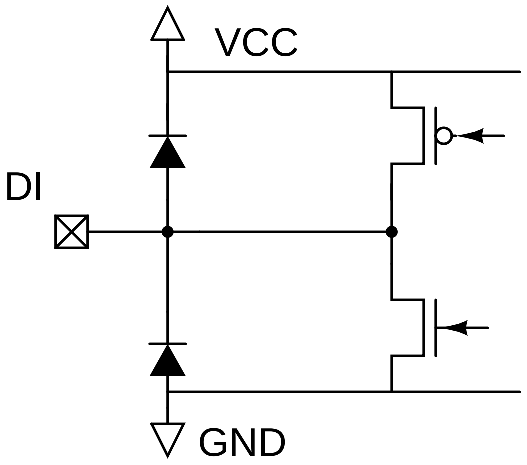


Figure 11: DI pin circuit

DI is only available with Tango C44. Figure 11: External Tango master to slave data. This output can drive a Tango F32 or F48 DI input pad.

1.2.10 CK

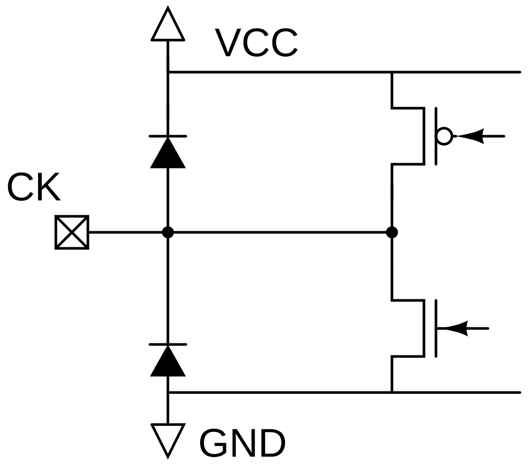


Figure 12: CK pin circuit

CK is only available with Tango C44. Figure 12: External Tango master clock. This output can drive a Tango F32 or F48 CK input pad.

1.2.11 DO

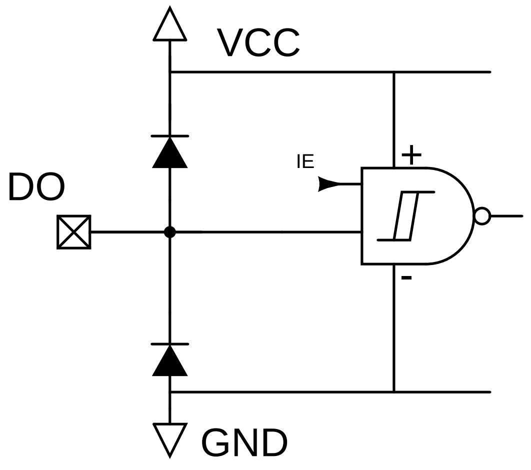


Figure 13: DO pin circuit

DO is only available with Tango C44. Figure 13: External Tango slave to master data. This input can receive signal from a Tango F32 or F48 DO output pad.

1.2.12 Internal Tango sensor line

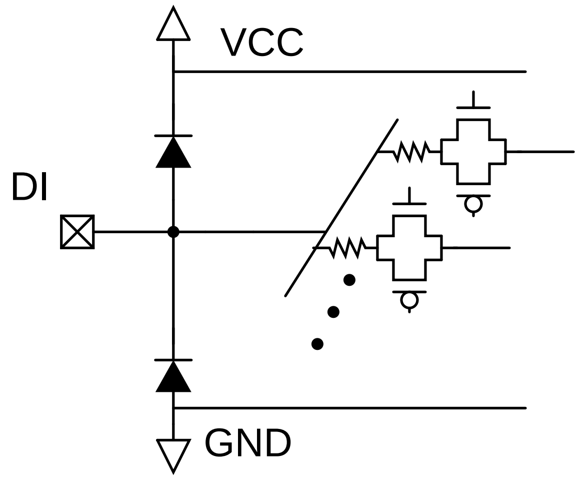


Figure 14: sense pin circuit

Figure 14: Tango C44 has 44 sensor lines(T$_{2}$ ... T$_{45}$). Tango C48 has 48 sensor lines(T$_{0}$ ... T$_{47}$), and Tango C32 has 32 sensor lines(T$_{0}$ ... T$_{31}$).

1.3 Absolute Maximum Ratings

Table 5: Absolute Maximum Ratings

Symbol	Parameter	Conditions	Unit
VCC	Power supply	-0.3 to 3.6	V
V$_I$	Input voltage	-0.3 to VCC + 0.3	V
V$_O$	Output voltage	-0.3 to VCC + 0.3	V
Top	Operation temperature	-30 to 85	ºC
Tstg	Storage temperature	-55 to 125	ºC
V$_{ESD}$	Electrostatic discharge (HBM model)	4’000	V

Stresses or exposure above these maximum ratings may cause permanent damages to the device.

MSL: Level 3

1.4 Recommended Operation Conditions

Table 6: Recommended Operation Conditions

Symbol	Parameter		Standard			Unit
			Min.	Typ.	Max.	
VCC	Power supply		2.5	3.3	3.6	V
GND	Power supply		-	0	-	V
V$_{IH}$	Input “H” voltage		0.8VCC	-	VCC	V
V$_{IL}$	Input “L” voltage		0	-	0.2VCC	V
I$_{OH}$(peak)	Peak output “H current	Driver capacity Low	-	-	10	mA
		Driver capacity High	-	-	40	mA
I$_{OL}$(peak)	Peak output “L” current	Driver capacity Low	-	-	10	mA
		Driver capacity High	-	-	40	mA
-	System clock frequency		-	-	20	MHz
-	CPU clock frequency		-	-	20	MHz
-	Operating temperature		20	-	75	ºC

1.5 Electrical interface

1.5.1 Interfacing with low supply host

The MSI interface has a minimum of 3 signals. The data communication between the MSI controller and the host uses 3 wires. An attention line, called ATTB, and the I^{2}C signals SDA and SCL, as shown in figure 15. The MSI controller is allowed to set the ATTB line .

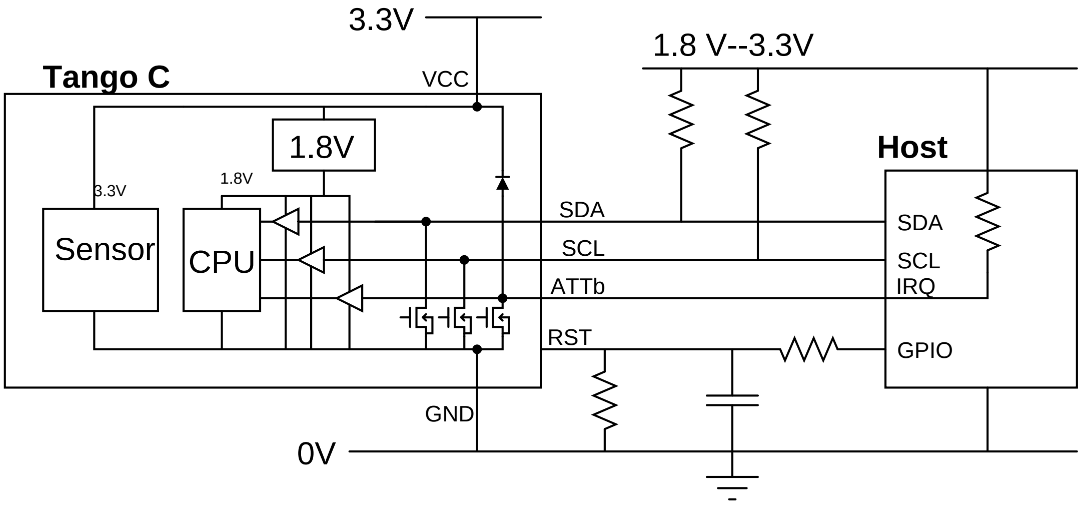


Figure 15: Example of interface with low voltage IIC bus

Also Tango Core have only a single, analog VCC supply, it generates an internal 1.8V supply for its digital part. When the low input thresholds are selected for the IIC pads SDA, SCL and optional ATTB, it is compatible with low voltage IIC bus. Use eeprom parameter 4.7.24EEP_IO_SWITCHING_LEVEL can swich voltage 1.8V or 0.9V.

1.5.2 States of ATTB line

The slave can set the ATTB line, and host can read and write MSI device. so the MSI controller behaves like an I^{2}C slave device and fully complies with I^{2}C addressing and usual I^{2}C hand shake protocol. As such, a MSI controller is suitable in an bus shared with other I^{2}C slaves.

1.5.3 Transition of ATTB line

When INT_MODE=00 in the INT mode register, the slave will set the ATTB line with INT_width pulse width after each scan in order to request the attention from the host, as shown in figure 16 and figure 17.

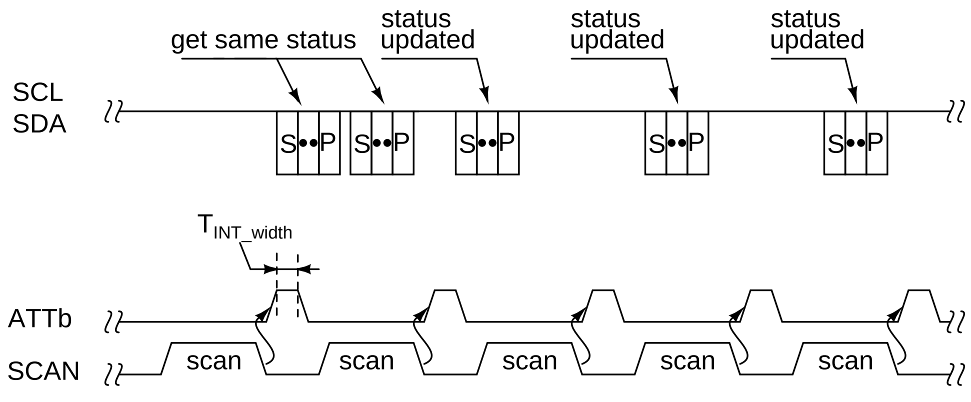


Figure 16: ATTB line pull up by slave (INT_POL=1,INT_MODE=00 in the INT mode register)

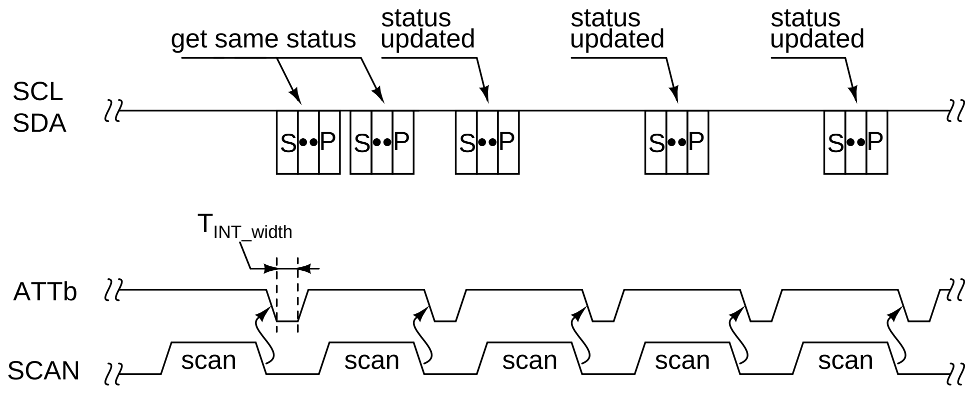


Figure 17: ATTB line pull down by slave (INT_POL=0, INT_MODE=00 in the INT mode register)

When INT_Mode=01 in the INT mode register and finger moving on the panel, the slave will set the ATTB line after each scan, as shown in figure 18. When finger leaves the panel, the slave will continue to pulse ATTB line for each scan; but once the master has serviced this request and become now aware that there is no more finger touching, the slave will stop pulse the ATTB line, and will also gradually reduce the scan speed,as shown in figure 19.

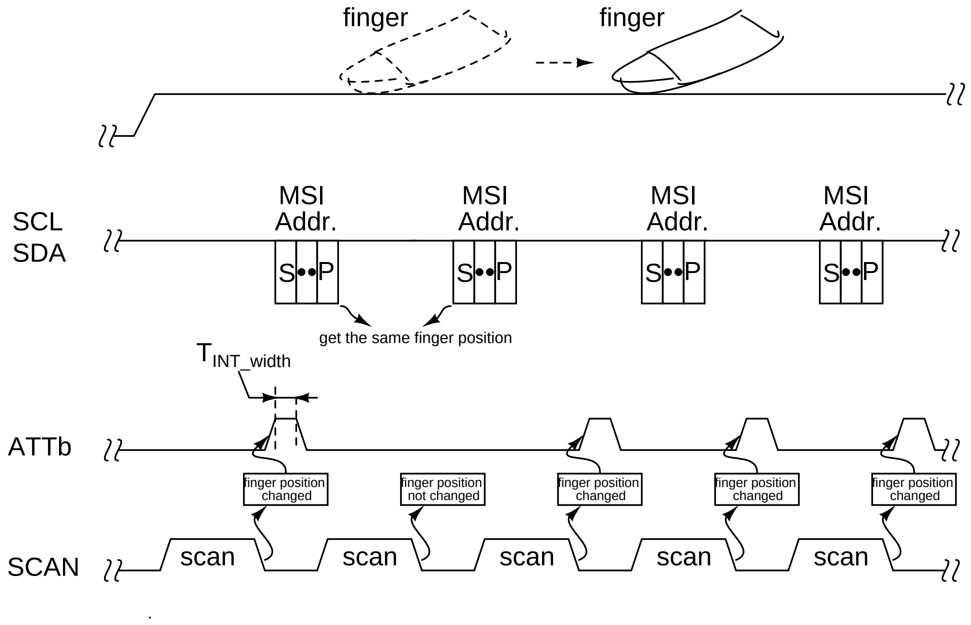


Figure 18: ATTB line pull up when finger moving (INT_POL=1, INT_MODE=01 in the INT mode register)

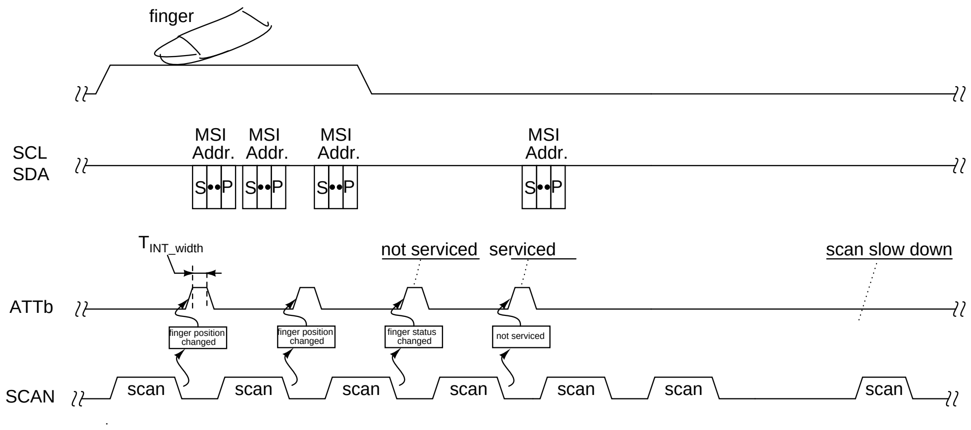


Figure 19: ATTB line will stop pulse when finger leaves and master has acknowledged the situation (INT_POL=1 in the INT mode register)

When INT_Mode=10 in the INT mode register and finger touch the panel, the slave will set the ATTB line after each scan, as shown in figure 20. When finger leaves the panel, the slave will continue keep ATTB line status for each scan; but once the master has serviced this request and become now aware that there is no more finger touching, the slave will release the ATTB line, and will also gradually reduce the scan speed,as shown in figure 21 .

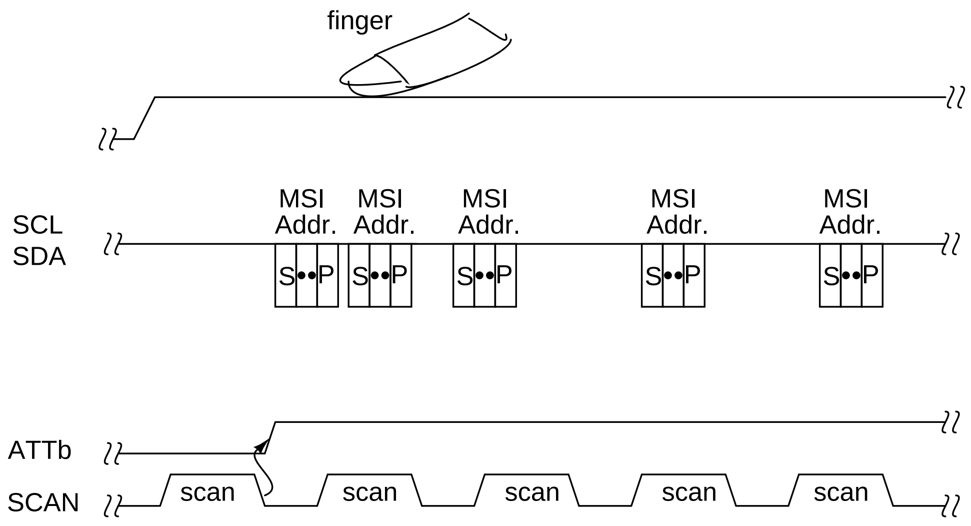


Figure 20: ATTB line pull up when finger touch (INT_POL=1, INT_MODE=10 in the INT mode register)

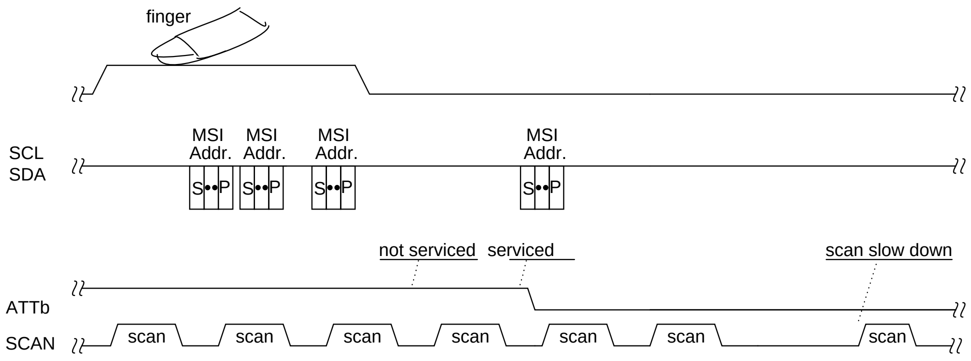


Figure 21: ATTB line will reset level when finger leaves and master has acknowledged the situation (INT_POL=1 in the INT mode register)

The only differece is send ATTB pluse instead of level between INT_Mode=10 to INT_Mode=11.

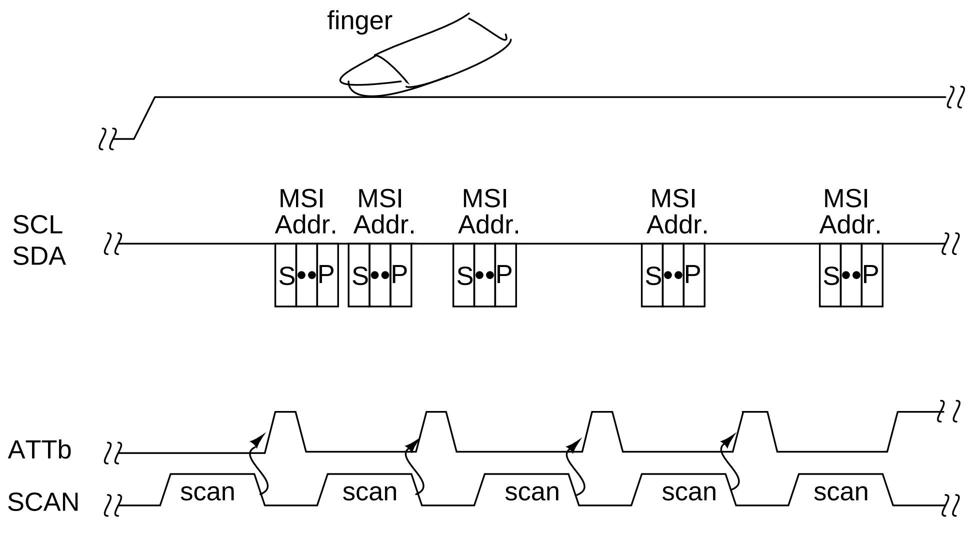


Figure 22: ATTB line pull up when finger touch (INT_POL=1, INT_MODE=11 in the INT mode register)

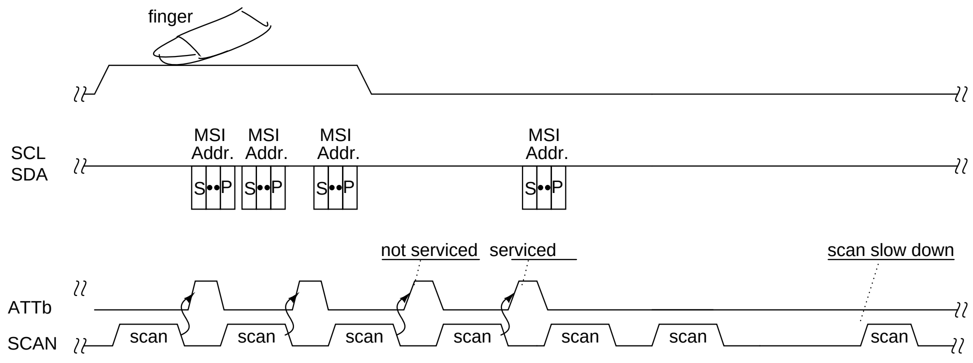


Figure 23: ATTB line will stop pulse when finger leaves and master has acknowledged the situation (INT_POL=1 in the INT mode register)

1.6 Schematic for C32/C44/C48/C44+F32 circuit

Pinmaps design ruler:

X axis: Tango Core default name "X axis" which have scan channles are more than other axis (**X axis>=Y axis**).

X asix and Y asix are used describe scanning sequence, does not conflict with customers choose coordinate origin.

Maximum scan pin: X axis <=48, Y axis <=32, X +Y <=75.

Buttons: Maximum 8 buttons, Minimum buttons is 2.

Dummy pin: At least one (X axis,Y and button dummy share with one dummy).

If have enough sensor lines, we suppose two dummy pins for each axis and one dummy for buttons.

Chip solution	Chip type	Total Sensors lines	Max scan pin for Active area	Dummy pin	button
Single Tango	C32	32	31(X+Y)	See above **Dummy pin**	See above **button**
	C48	48	47(X+Y)		
Twin Tango	C44+F32	76	75(X+Y)		

Table 7: pimmap design ruler

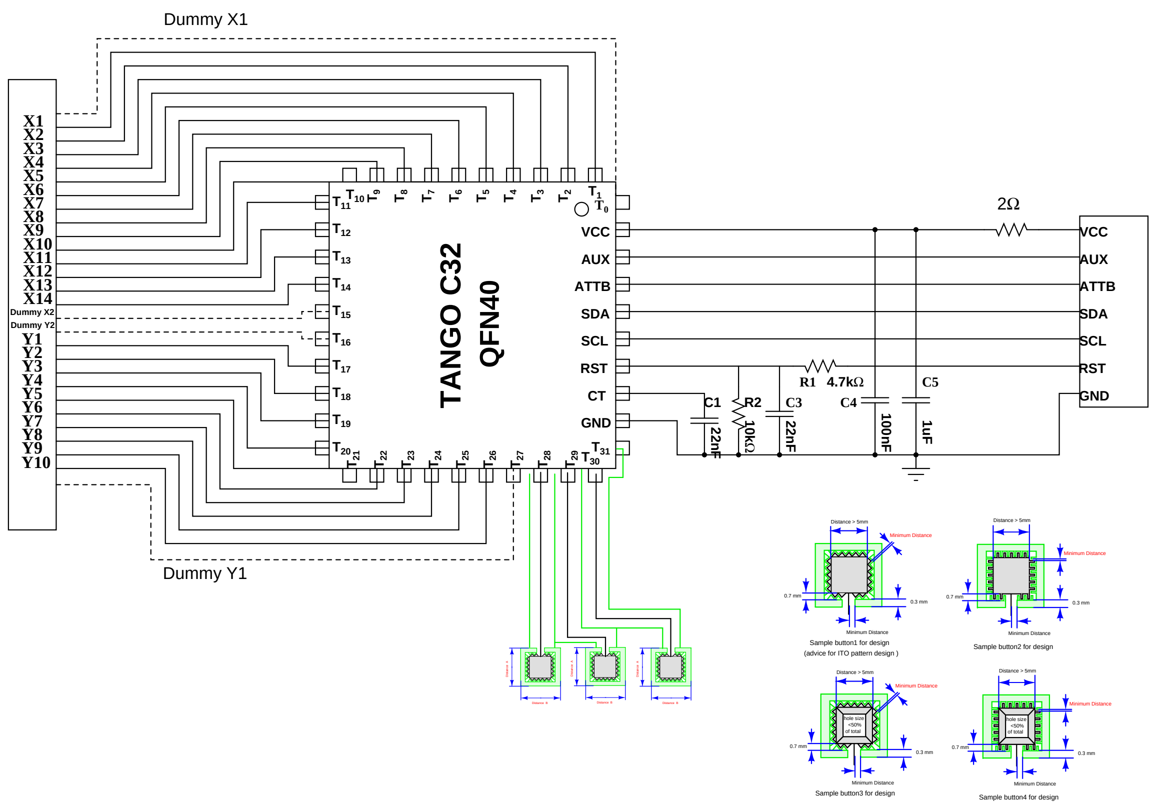


Figure 24: tango_C32_circuit

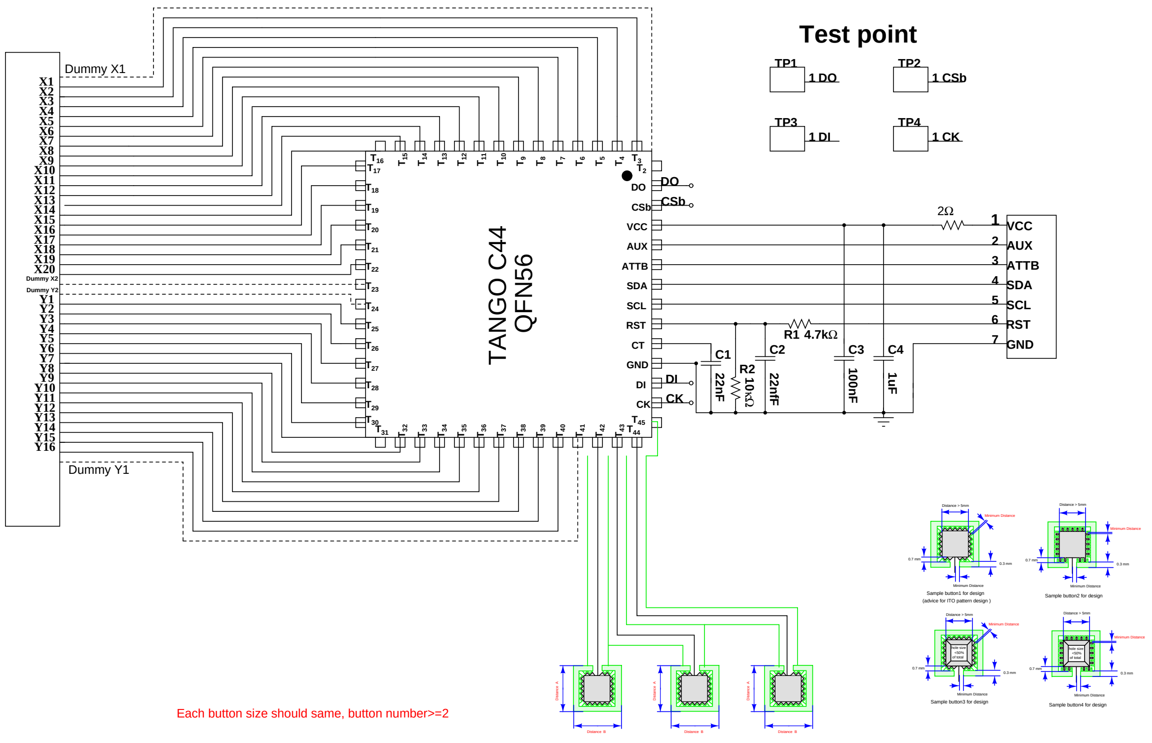


Figure 25: tango_C44_circuit

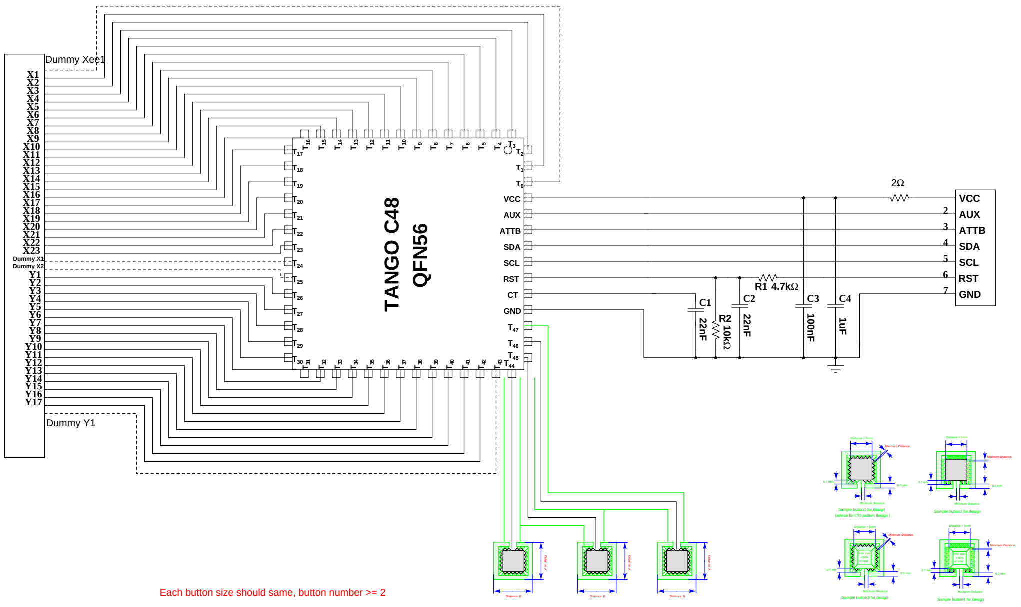


Figure 26: tango_C48_circuit

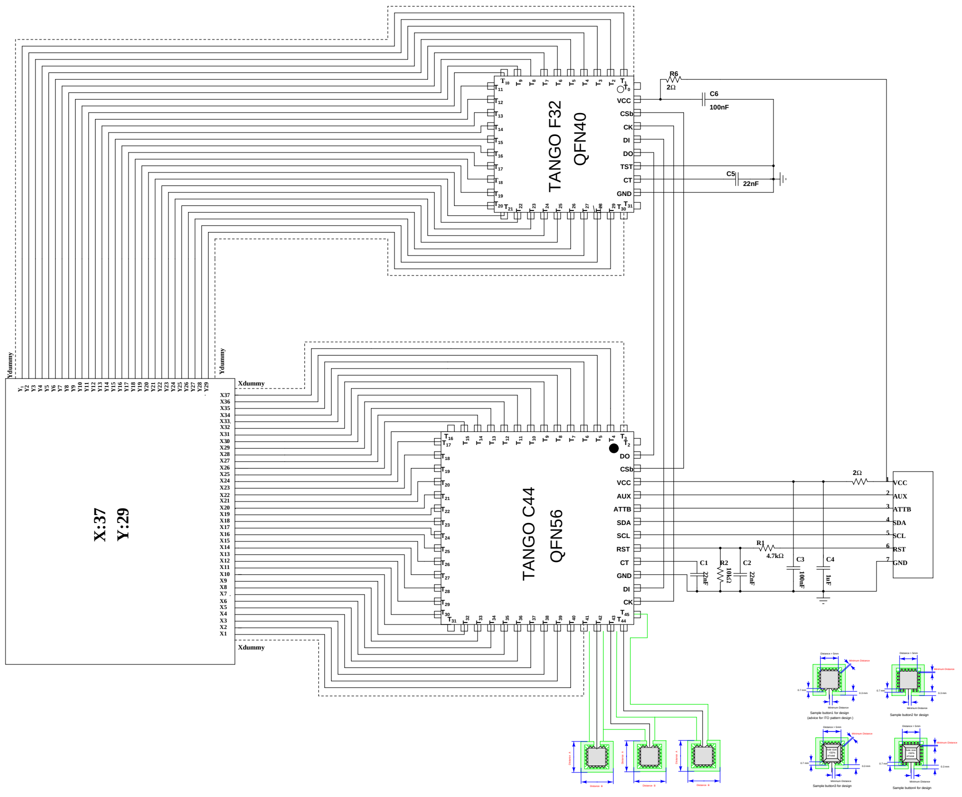


Figure 27: tango_C44+F32_circuit

1.7 Reset sources and sequences

The specification of the reset and power modes are summarized in tables 8, 9, 10.

1.7.1 Power on reset

Reset pin tied to ground with 10kΩ resistance, result shown in figure 28.

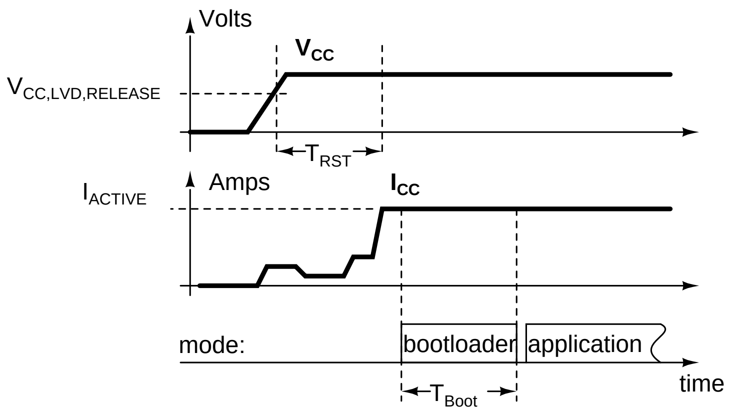


Figure 28: Power on sequence-1

Table 8: Power on reset

Symbol	Parameter	Min	Typ	Max	Unit
V$_{CC,POR,}$	RISE Power supply rise time for correct power on reset	1000			V/s
T$_{POR,REGEN}$	Power on reset regeneration time		5		us
V$_{CC,POR,REGEN}$	Power on reset regeneration level, V$_{CC}$must go below V$_{CC,POR,REGEN}$in order to regenerate the POR, after POR has turned off		tbd		V
T$_{POR,OFF}$	Power on reset turn off time	0.4		0.7	ms

Table 9: Low level detector

Symbol	Parameter	Min	Typ	Max	Unit
V$_{CC,LVD,SET}$	Low level detector set threshold, going lower will reset		2.3		V
V$_{CC,LVD,RELEASE}$	Low level detector release threshold		2.42		V
T$_{LVD}$	Minimum VCC drop duration below VCC,LV D,SET ensuring a reset		tdb		s

Table 10: Power consumption, for the CPU

Symbol	Parameter	Min	Typ	Max	Unit
I_{PROG}	Flash programming current			7	mA
I_{ERASE}	Flash erasing current			7	mA
I_{MERASE}	Flash mass erase current			7	mA
I_{ACTIVE}	Active mode dynamic consumption, no touch sensor connected, VCC=3.3V, 20MHz CPU		6	9	mA
I_{WAIT}	Wait mode, no touch sensor connected, VCC =3.3V, 20MHz CPU		3		mA
I_{SLEEP}	Sleep mode, no touch sensor connected, VCC =3.3V, 20MHz CPU. Consumption includes periodical wake up: 60ms sleep and 50us active execution (1000 cycles).		45		uA
I_{FREEZE}	Freeze mode, no touch sensor connected, VCC =3.3V, 20MHz CPU		1		uA
I_{RESET}	Reset mode, no touch sensor connected, VCC =3.3V, 20MHz CPU		2		mA

1.7.2 Reset by RST pin

Two cases are distinguished:

1. Reset while Tango C is in active mode, shown in figure 30.
2. Reset while the Tango C is in sleep mode, shown in figure 32.

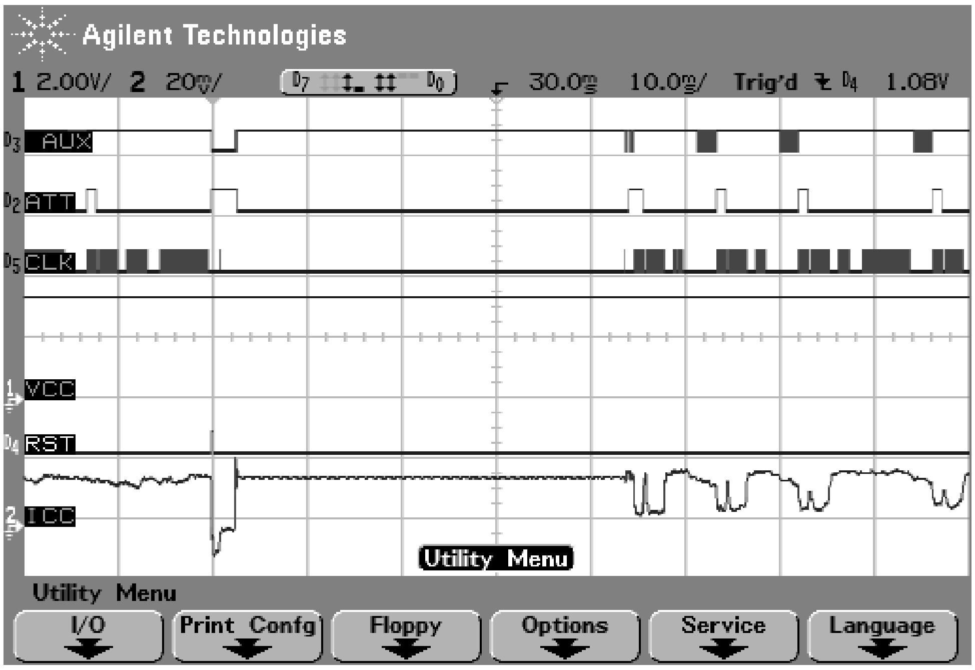


Figure 29: The oscilloscope shows the rest when in active

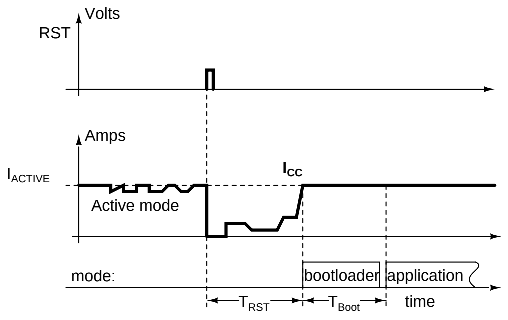


Figure 30: Power on sequence-2 (reset controlled by Host MCU when in active)

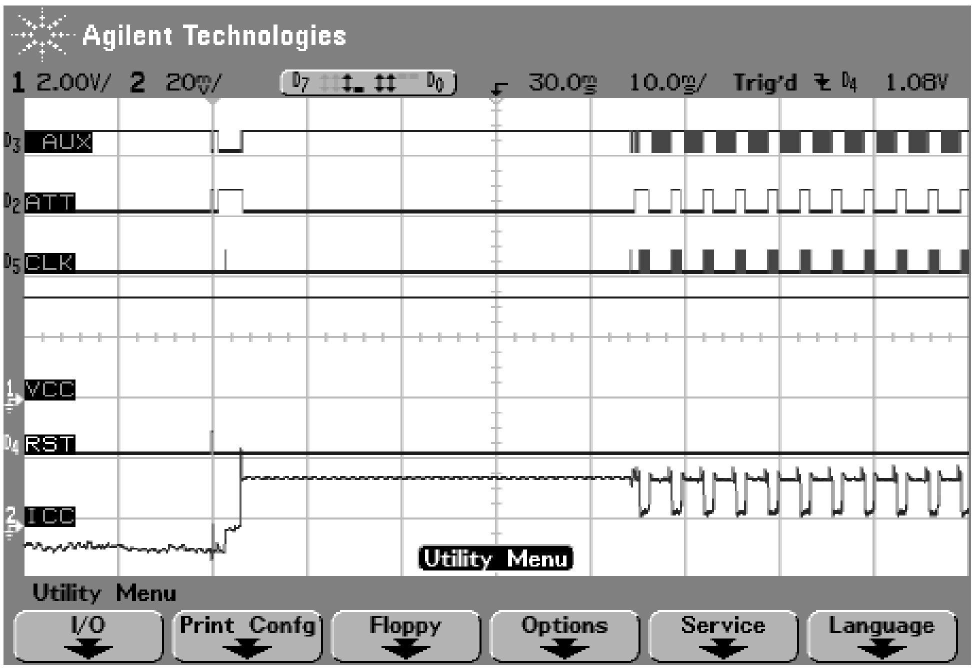


Figure 31: The oscilloscope shows the rest when in sleep

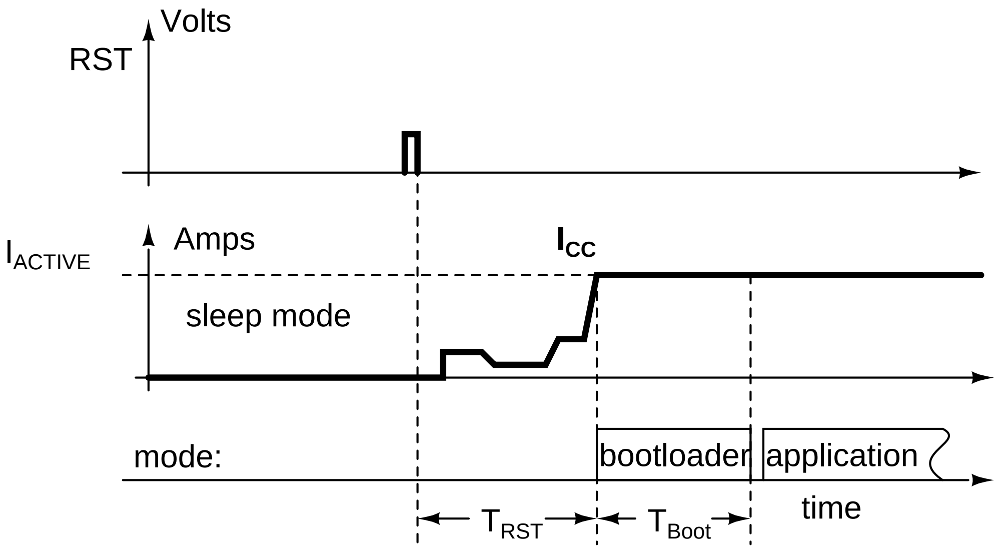


Figure 32: Power on sequence-3 (reset controlled by Host MCU when in sleep)

1.8 I^{2}C Protocol Specifications

1.8.1 I^{2}C features

1. Supports a baud rate up to 400kHz (Fast Mode).
2. Only support single master solution.
3. Only support 7-BIT addressing.
4. If I^{2}C master can't finish 1byte data in 100ms, I^{2}C slave will restart. Tango C44 operates only as a slave device.The I^{2}C interface is functional in active and sleep modes. In sleep mode, an asynchronous address match detector hardware allows a sleeping Tango C44/C48 to recognize its address and wake up. And The firmware can implements different I^{2}C touch protocols.The timings for example that as table 11 and figure 33.
5. I^{2}C slave can hold off the master in the middle of a transaction using what's called clock stretching (the slave keeps SCL pulled low until it's ready to continue). Refer to figure 34 for a example.

1.8.2 I^{2}C address

1. MSI device address = 0x5C.

Table 11: I^{2}C timing

Symbol	Parameter	Min	Typ	Max	Unit
T_{LOW}	IIC clock low time	$2 \bullet T_{CPU}$			
T_{HIGH}	IIC clock high time	$2 \bullet T_{CPU}$			
$T_{HD,STA}$	IIC clock hold time	$2 \bullet T_{CPU}$			
$T_{SU,STA}$	IIC start setup time				
$T_{SU,STO}$	IIC stop setup time				
$T_{HD,DAT}$	IIC data hold time, when driven by master side				
$T_{SU,DAT}$	IIC data setup time, when driven by master side				
T_{BUF}	IIC bus free time	4.7			us
$T_{CS,A}$	IIC clock stretching timer before acknowledge from slave side, when CPU is in CPU_active or CPU_wait mode		tbd		
$T_{CS,S}$	IIC clock stretching timer before acknowledge from slave side, when CPU is in CPU_sleep mode		tbd		
T_{CSR}	IIC clock stretching release time	$9 \bullet T_{CPU}$			
$T_{VD,DAT}$	I C data valid after clock change, when data is driven by slave side	$9 \bullet T_{CPU}$			
T_{TCPU}	CPU master clock period			55	ns

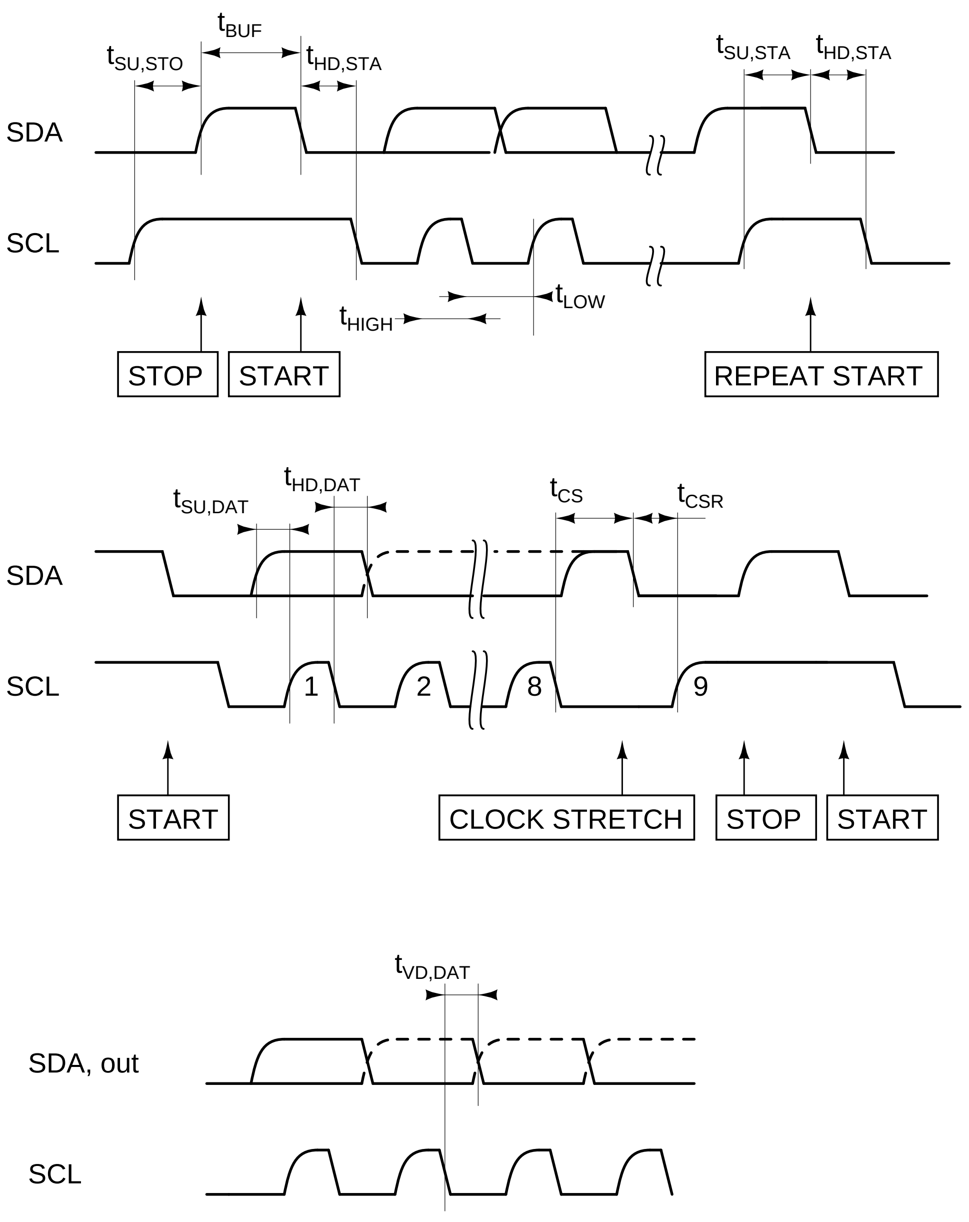


Figure 33: I^{2}C timing example

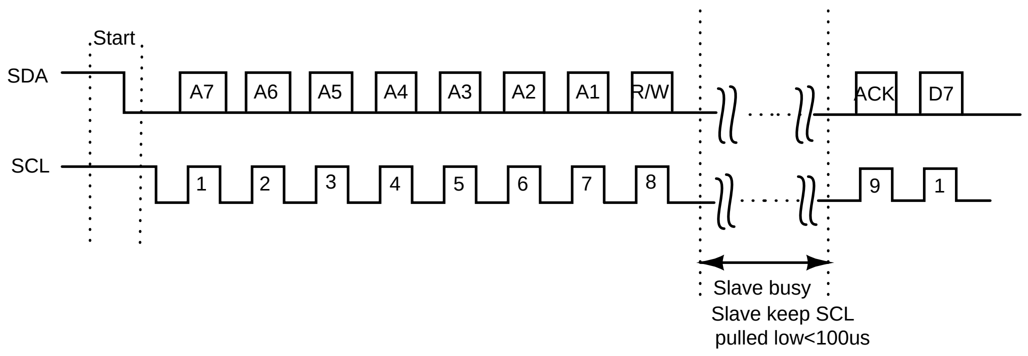


Figure 34: I^{2}C clock stretching example

2 Data Protocol

The communication follows I^2C convention. Refer to figure 35 for a definition of the symbols used.

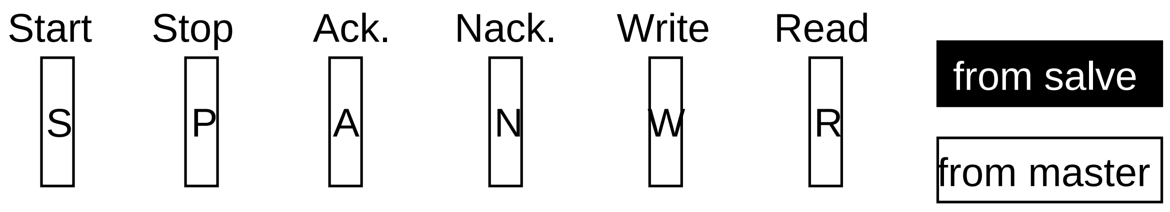


Figure 35: I^2C symbols

2.1 Introduction

The protocol for data exchange has been designed with the following considerations

- Most of the data traffic are read operation to get the finger or fingers position
- Read operation do need an initial write operation.
- Write operations are most of the time power management and interrupt setting instructions
- Interrupt pulse width setting adjustments need a write operation.

2.1.1 Read operation

Read packets have variable content length, decided by the host. It is available to do a single read operation or a sequential read operation. Therefore, the beginning register address is need to set before a read operation. And the data sent exactly follow the register table 12, table 14, table 15, table 18. And,the firmware in the slave will use a memory copy of the register for I^2C slave read operation, so that firmware can continue updates, and I^2C slave is still using a consistent (but old) coordinates for read operation.

- In a sequential read operation, the first data sent by the MSI device is therefore the touching register, and then the X and Y coordinates of the first finger, then 2nd finger, 3rd finger, 4th finger and then coordinates of the 5th finger, and so on. referred in figure 37.
- If the host do not finish the read operation when the ATTB line is set again, the slave firmware will delay to update coordinates registers for I^2C read operation until the host finish the read operation. referred to first part of figure 38.
- I^2C stop condition will release data protection and allow the slave firmware update the coordinates registers for I^2C read operation. So, the host has the chance to get incorrect data when it get the coordinates data with single read operation. Because the host send many times of I^2C stop condition in each multi-fingers coordinates position reading, it will give the slave firmware chance to update the coordinates registers for I^2C read operation, the host will give a combines unrelated data (combines new and old coordinates together), referred to the second part of figure 38.

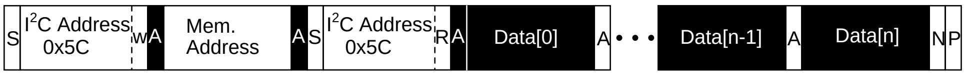


Figure 36: Read operation

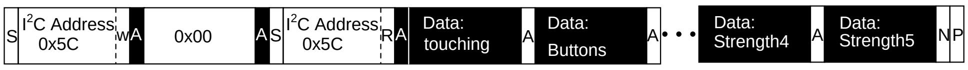


Figure 37: Coordinates read operation

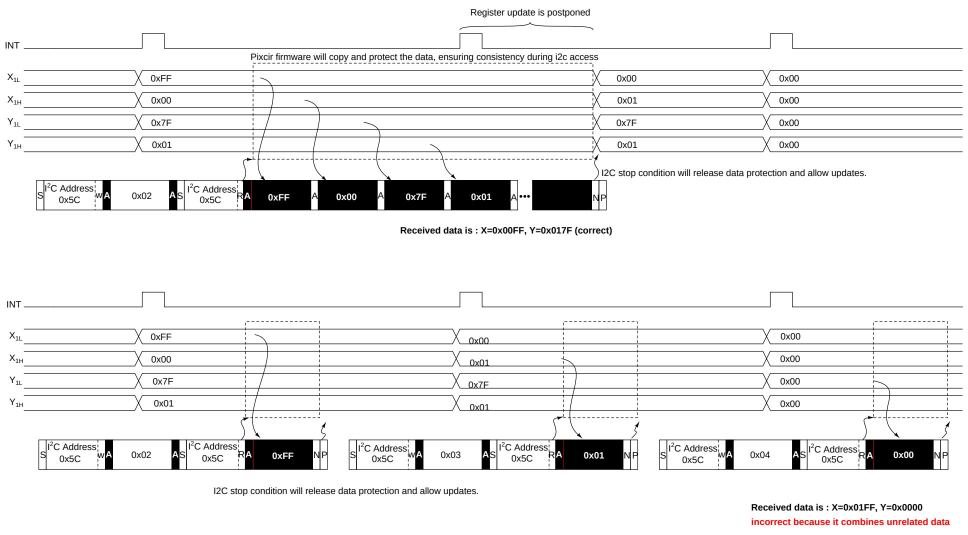

Register update is postponed
INT
Pixcir firmware will copy and protect the data, ensuring consistency during i2c access
X1L
X1H
Y1L
Y1H
0xFF
0x00
0x7F
0x01
I2C stop condition will release data protection and allow updates.
I2C Address 0x5C
0x02
0xFF
0x00
0x7F
0x01
Received data is : X=0x00FF, Y=0x017F (correct)
0x03
0x04
I2C stop condition will release data protection and allow updates.
Received data is : X=0x01FF, Y=0x0000
incorrect because it combines unrelated data

Figure 38: Coordinates read operation explanation

2.1.2 Write operation

Write packets have variable content length, decided by the host. Write operation stops when host issues an I^{2}C STOP symbol. The write packet is illustrated in figure 39 and figure 40. Following the I^{2}C device address, the first byte of the write packet is always the destination register address, referred in table 12, table 14, table 15, table 18. Subsequent data value are written at the register pointed by the address, immediately upon reception of the byte. The address counter is automatically incremented. Subsequent data bytes are treated in continuation of the writing operation.

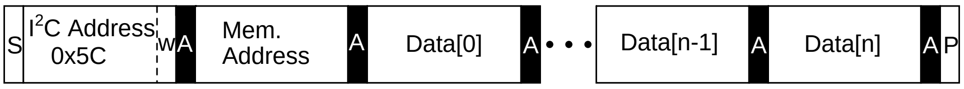


Figure 39: Write operation.

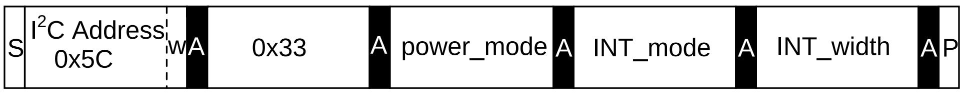


Figure 40: Write mode setting operation .

2.2 MSI Registers

2.2.1 Endianness

Data are little endian, which means LSB byte appears before MSB byte.

2.2.2 Registers organization

The accessible registers are shown in the table 12, table 14, table 15, table 18. These registers are technically accessible both for reading or writing direction. However, most registers have only one meaningful direction: finger position registers, for example, are typically used in read direction, and writing to them will have no effect; their content will be overridden after a new sensor scan.

Table 12: MSI registers, coordinates

Address	Type	Name	Description	Category
0	char	`touching`	Bit field, see table 13	touch
1	char	`buttons`	Buttons bitfield	
2 (lsb) 3 (msb)	int	`posx1`	Finger #1 X position	
4 (lsb) 5 (msb)	int	`posy1`	Finger #1 Y position	
6	char	`id1`	Finger #1 identificator	
7 (lsb) 8 (msb)	int	`posx2`	Finger #2 X position	
9 (lsb) 10 (msb)	int	`posy2`	Finger #2 Y position	
11	char	`id2`	Finger #2 identificator	
12 (lsb) 13 (msb)	int	`posx3`	Finger #3 X position	
14 (lsb) 15 (msb)	int	`posy3`	Finger #3 Y position	
16	char	`id3`	Finger #3 identificator	
17 (lsb) 18 (msb)	int	`posx4`	Finger #4 X position	
19 (lsb) 20 (msb)	int	`posy4`	Finger #4 Y position	
21	char	`id4`	Finger #4 identificator	
22 (lsb) 23 (msb)	int	`posx5`	Finger #5 X position	
24 (lsb) 25 (msb)	int	`posy5`	Finger #5 Y position	
26	char	`id5`	Finger #5 identificator	
27	char	`strength1`	Finger #1 strength	
28	char	`strength2`	Finger #2 strength	
29	char	`strength3`	Finger #3 strength	
30	char	`strength4`	Finger #4 strength	
31	char	`strength5`	Finger #5 strength	

Table 13: Touching

Bit 0,1,2	Nb of fingers touching (NBF)
Bit 3	Noise flag (indicates the report is unreliable) (NOI)
Bit 4	message flag (indicates a message string is sent by slave) (MSG)
Bit 5	buffer indicates the master has missed more than 2 reports, which are stored in buffer array (BUF)
Bit 6	palm flag (indicates the algorithm has a palm or similar blocking issue) (PAL)
Bit 7	water flag, indicates the algorithm has a rejected inputs due to water (WAT)

Table 14: MSI registers, gestures and monitoring

Address	Type	Name	Description	Category
32 (lsb) 33 (msb)	int	initial_distance	Distance separating fingers on the first time multitouch is detected	gesture
34 (lsb) 35 (msb)	int	distance	Distance separating fingers	
36 (lsb) 37 (msb)	int	ratio	100·distance / initial_distance	
38	char	water_level		monitor
39	char	noise_level		
40	char	palm_level		
41	char	signal_x		
42	char	signal_y		
43 50	char	button1button8	Signal level of the buttons	buttons
51	char	power_mode	Power management register. See subsection §2.2.4 and table 19	power management
52	char	INT_mode	Control of the ATTB pin, see subsection §2.2.5 and table 20	
53	char	INT_width	ATTB pulse width	
54	char	Sleep_freq	Scanning frequence in Sleep mode	
55	char	Auto_sleep_delay	The delay time, the start is the last touch released in Active mode and the end is switch into Sleep mode succesful	
56-57	char		reserved for future use	special operations
58	char	SPECOP	Special operation . See table 16	
59 (lsb) 60 (msb)	int	EEPROM_read_addr	Address used during special operation	
61	char	Engineering_cmd	Allows, with I^{2}c, to send "hyper terminal like commands" for engineering modes	
62 (lsb) 63 (msb)	int	CRC	FLASH CRC value (must be requested by SPECOP), excluding "EEPROM" zone	version
64-95	char	version[0..31]	Customer version control (32bytes) (imap to "eeprom")	

Table 15: MSI registers, gestures and monitoring

Address	Type	Name	Description	Category
96-135	char	message[0..39]	Null terminated ASCII message string for engineering and debug purpose	
136 (lsb) 137 (msb)	int	RAW_CTRL	Controls RAW data mode (internal, raw, etc. . .) see table 17	
138	char	cross_x	X coordinate for method 1 crossing node measurement request	method 1
139	char	cross_y	Y coordinate for method 1 crossing node measurement request	
140 (lsb) 142 (msb)	int	cross_node	Measurement result for method 1	
142 (lsb) 143 (msb) 144 (lsb) 145 (msb) etc.	int int int	RAW[0..69] shared with history_buffer	Raw data, content controlled by RAW_CTRL register, or alternatively, history buffer (see below)	RAW data

Table 16: SPECOP

0	Normal operation
1	"EEPROM" read operation, start address must be written in EEPROM_read_addr
2	"EEPROM" write operation NOT IMPLEMENTED
3	Calibration
4	CRC checksum of the application in Flash

2.2.3 RAW_CTRL write&read

It is advised to use INT mode=0x08 when debug information are consulted (RAW_CTRL register not zero). Also, the slave can not instantly refresh the RAW tables following a modification by the master to the RAW_CTRL register, since in some conditions a relatively lengthy collection of measurements has to be performed. The master however can have the guaranty that the data reported in the RAW table reflects the request placed in RAW_CTRL if 2 ATTb pulses have elapsed. If the request in RAW_CTRL is unchanged, to every new ATTb pulse corresponds a refresh of the RAW table.

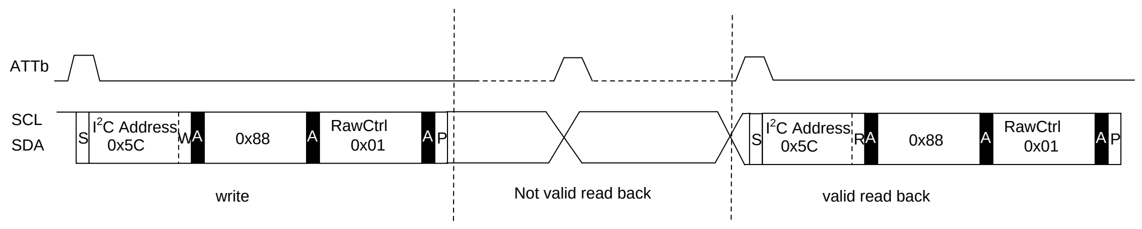


Figure 41: RAW_CTRL write&read

Table 17: RAW_CTRL

Bit 0	Choose function (0: history buffer, 1: RAW data, 2: system info) See table 18
Bit 1	
Bit 2	Method (0 0r 1)
Bit 3	Show offset correction (and low-pass filtering for M0)
Bit 4	Show m0 sensitivity adjustment (bit3 must also be set)
Bit 5	M1 pattern small (0) or pattern large (1)
Bit 6	M1 sense direction (0:Y,1:X)
Bit 7	M1 band scan. if 0, only report a single cross node. If 1,report a full X axis scan at RAW position
Bit 8	Disable Algorithm
Bit 9	Enable single shot RAW refresh, must be set to 1 and bit9 to 0. Auto back to 0 and bit9 to 1 after single shot is done
Bit 10	Refresh frozen after single shot is done when 1. Set to 0 to release the freeze and go back to normal refreshing
Bit 11	
Bit 12	
Bit 13	
Bit 14	
Bit 15	

Table 18: MSI registers, history buffer

Address	Type	Name	Description	Category
142	char	interval	Sub sampling rate when filling the history buffer. Disable: 0. Keep all points: 1. Keep one out of two: 2. Etc.	history buffer
143	char	buffer_level	Number of fingers report in the buffer	
144 (lsb) 145 (msb)	int	posx	Coordinate X of the reported point, at time=0	
146 (lsb) 147 (msb)	int	posy	Coordinate Y of the reported point, at time=0	
148 (lsb) 149 (msb)	int	posx	Coordinate X of the reported point at time=1	
150 (lsb) 151 (msb)	int	posy	Coordinate Y of the reported point at time=1	
...				
298 (lsb) 299 (msb)	int	posx	Coordinate X of the reported point, at time=19	
300 (lsb) 301 (msb)	int	posy	Coordinate Y of the reported point, at time=19	

2.2.4 Power_mode register

The POWER_MODE register controls the power management and operation of the MSI device. However, modification becomes effective at any time.There are shown in the table 19

Table 19: Power_mode register

Bit	Name	Description
7-3	-	Not used
2	ALLOW_SLEEP	Allow self demotion from active to sleep mode, provide that this flag is set. If the MSI device is in active mode and no finger is detected for more than IDLE_PERIOD time, then it allow automatically jumps to sleep mode. If this flag is not set, the host must explicitly switch the device from active to sleep mode.
1-0	POWER_MODE[1-0]	Power mode setting of the MSI device: 00: Active Mode 01: Sleep Mode 11: Freeze Mode

2.2.5 INT_mode register

Table 20: INT_mode register

Bit	Name	Description
7-4	-	Not used
3	EN_INT	0: disable interrupt mode 1: enable interrupt mode
2	INT_POL	0: the interrupt is low-active (default) 1:the interrupt is high-active
1-0	INT_MODE[1-0]	00: INT assert periodically 01: INT assert only when finger moving(default) 10: INT level assert only when finger touch 11: INT pulse assert only when finger touch

2.3 Details about coordinates report and finger ID

2.3.1 Double buffering and atomic access

The I^{2}C register table is a resource which is potentially accessed by two different asynchronous processes. After each new scan of the sensor, touch coordinates are computed, and the values in the I^{2}C table are refreshed. Concurrently, the I^{2}C master may consult the I^{2}C registers table and read one byte after one byte the coordinates, a process whose speed is limited by the I^{2}C baud rate. It is important that the action of refresh, performed after the sensor scan, does not collide with the access by the I^{2}C master. Otherwise, the I^{2}C master may retrieve inconsistent data which is a mix of older (before refresh) and newer coordinates (after refresh), yield to large apparent touch errors. Pixcir implements a protection mechanism which ensures that data access by the I^{2}C master can not be cut (atomic) for the full time of the access. The mechanism is based on a system of double RAM buffer shown in figure 42. When the I^{2}C master starts a read operation of the slave, the section of registers containing finger coordinates is locked. Even if the section is locked, refresh operation is not put on hold, but is rather directed to a second RAM buffer. Keeping the refresh action on hold would reduce the scan rate. Once the I^{2}C bus is released, an instant buffer swap can be performed, presenting the freshest set of coordinates.

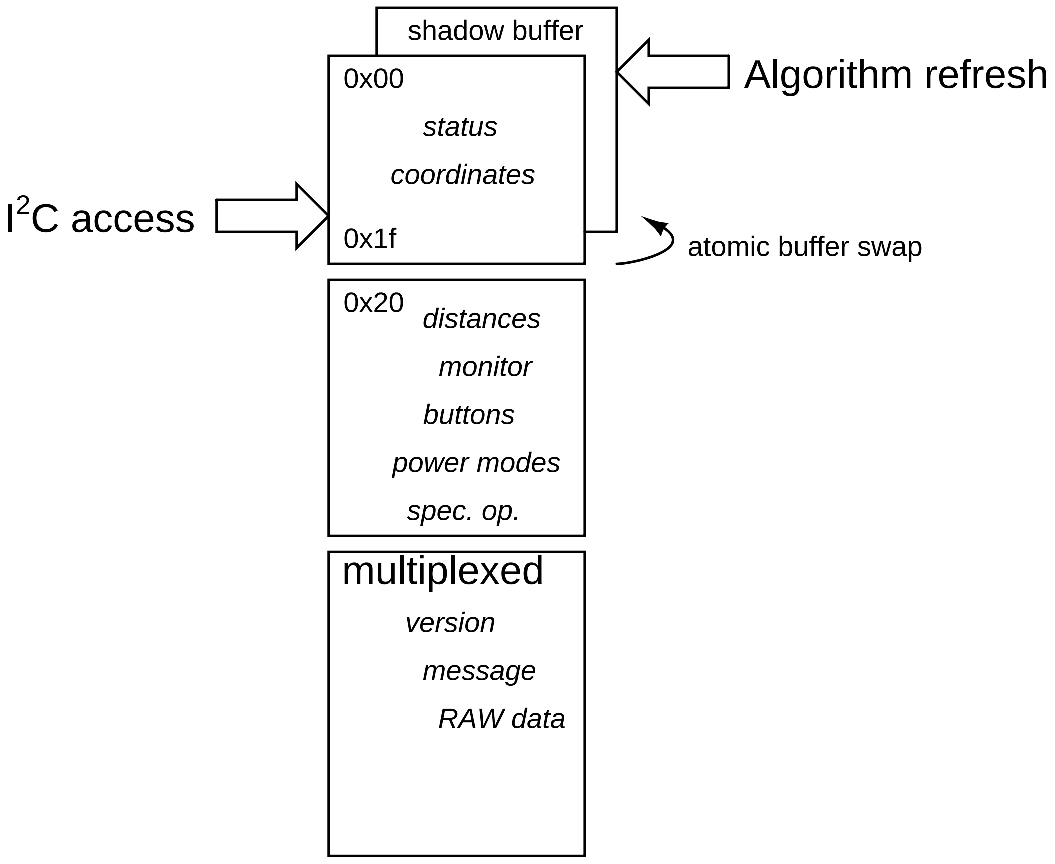


Figure 42: Registers table overview

The conditions for the mechanism to work are the following:

- The read operation of the coordinates table must be performed with a single I^{2}C transaction, starting with a single START and ending with a single STOP, potentially reading all 32 bytes continuously. It is forbidden to read the table in small parts interleaved with STOP and START or RESTART since a RAM refresh can potentially occur in between

- The read operation of the coordinates table must be performed within a limited time. The read operation should not last longer than 2x the scan time of the sensor: the first refresh action can be safely stored in the shadow buffer, but the subsequent refresh would overwrite the active I^{2}C buffer, and the master would retrieve a mix of data. One scan time is approximatively 4ms, therefore the I^{2}C read must finish within approximatively 8ms. Reading 32 bytes with an I^{2}C speed of 100bit/s takes normally less than 3ms.
- It is mandatory to release the I^{2}C bus as soon as the coordinates data are read by the master with a STOP.

2.3.2 Status registers

There are 2 status registers, TOUCH and BUTTON, respectively for fingers and buttons reports. The TOUCH register is described in table 21. One important value stored in TOUCH register is NBF (number of fingers)

Table 21: TOUCH register

register	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
TOUCH	WAT	PAL	BUF	MSG	NOI	NBF		

WAT Water flag, indicates report is unreliable; the level is reported in WATER_LEVEL

PAL Palm flag, indicates report is unreliable; the level is reported in PALM_LEVEL

BUF Buffer flag indicates that the master has missed more than 1 report; the unread coordinates are stored in a buffer mechanism explained in its dedicated section

MSG Message flag, indicates that the algorithm must display an ASCII string; explained in its dedicated section

NOI Noise flag, indicates the report is unreliable; the level of noise is reported in NOISE_LEVEL

NBF Number of fingers touching; ranging between 0 and 5 touching fingers

2.3.3 Difference in finger ID and finger slot

Fingers coordinates are reported using *report slots* in the I^{2}C table. These slots contain X, Y and ID information. In addition, the finger's touching strength is stored at the end of this section. *It is very important to understand the difference between report slots and finger ID*. When there is a number of touching fingers smaller than the maximum of 5 allowed, which means NBF<5, some report slots are unused. The I^{2}C master only needs to read the NBF first slots, since the firmware guaranties that unused slots are always at the end of the table. This reduces the amount of I^{2}C data. On the other hand, the order of the slots used for touching finger can not always be garantied. For example when the number of touching fingers decreases, empty slots could potentially appear in the middle of the table, and the algorithm will reorder the slots; this situation is shown at letter 'D' of the figure 44. Two coordinates reports contiguous in time will not necessarily allocate the same slots for the same fingers. It means that I^{2}C master should never rely on the report slot position as a replacement of finger ID. Only the finger ID is the valid information to keep track of fingers.

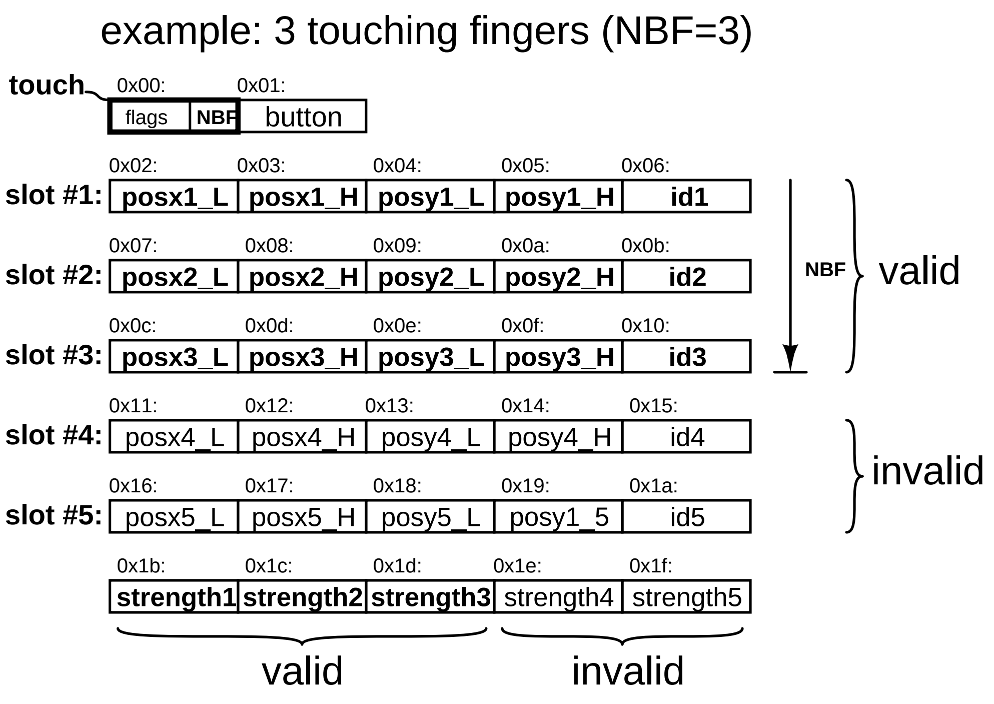


Figure 43: Coordinates section

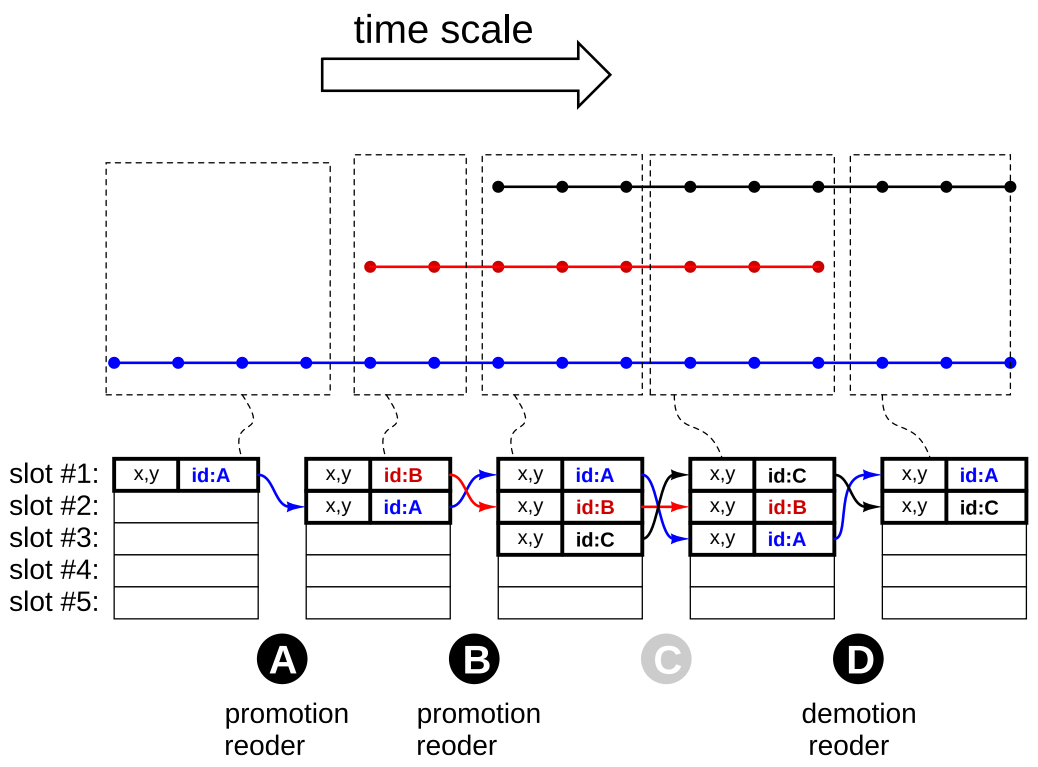


Figure 44: Finger slot system. Slot order is never guarantied, the master must always check the finger ID tag. Empty slots are however always at the end of the i^{2}c table.

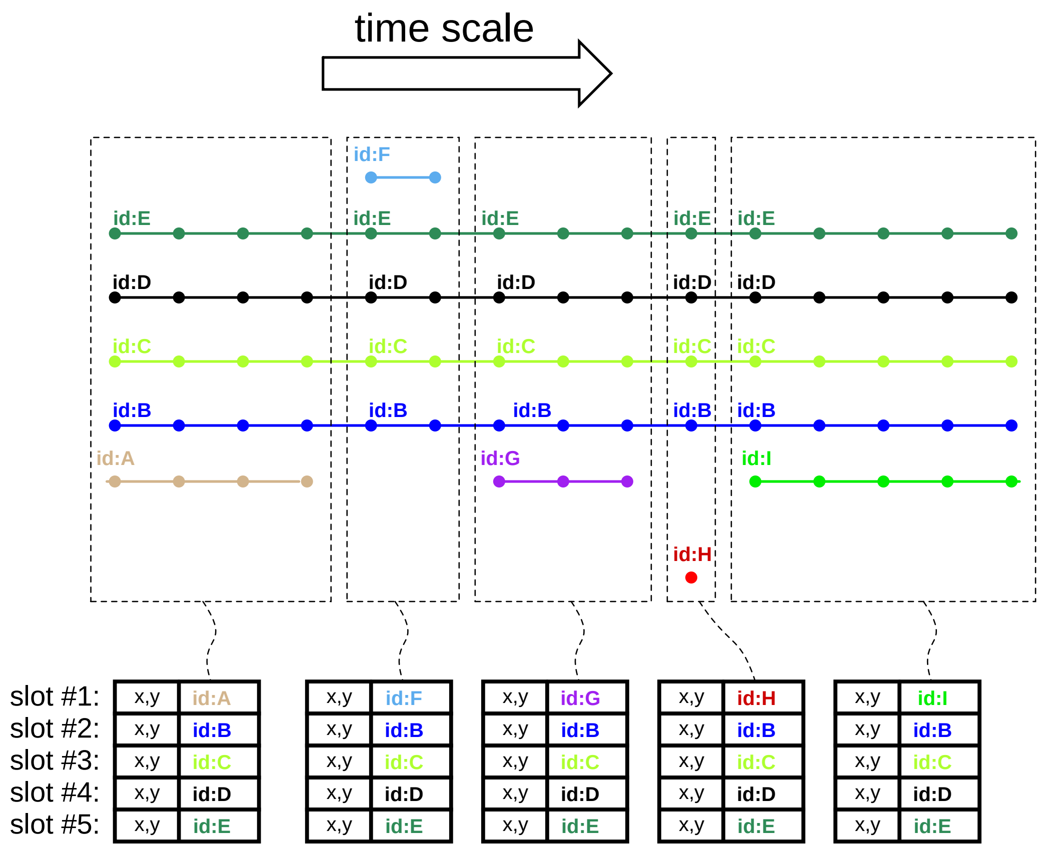


Figure 45: Finger ID system. Dots which can not link to an older report get a new finger ID assignement.

Algorithm 1 Fingers table

```
typedef struct
{
uint8_t touch;
uint8_t button;
uint8_t posx1_L;
uint8_t posx1_H;
uint8_t posy1_L;
uint8_t posy1_H;
uint8_t id1;
uint8_t posx2_L;
uint8_t posx2_H;
uint8_t posy2_L;
uint8_t posy2_H;
uint8_t id2;
uint8_t posx3_L;
uint8_t posx3_H;
uint8_t posy3_L;
uint8_t posy3_H;
uint8_t id3;
uint8_t posx4_L;
uint8_t posx4_H;
uint8_t posy4_L;
uint8_t posy4_H;
uint8_t id4;
uint8_t posx5_L;
uint8_t posx5_H;
uint8_t posy5_L;
uint8_t posy5_H;
uint8_t id5;
uint8_t strength1;
uint8_t strength2;
uint8_t strength3;
uint8_t strength4;
uint8_t strength5;
} PIXCIR_I2C_shadow_t;
```

2.4 Power management

There are four power modes. The master can change the power mode by writing into the slave POWER_MODE register. This operation can be executed at any time.

2.4.1 Active mode

In this mode, the slave resumes with a new scan directly after each I^{2}C transfer (after ATTB rising edge). This is used to reach the highest refresh rate (reach to 400Hz), but also has the highest current consumption. Figure 46 shows how to force the slave into Active mode.

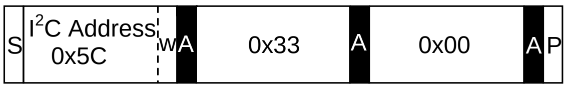


Figure 46: Active mode sequence

2.4.2 Sleep mode

This mode is selected to decrease the current consumption during low activity phases on the sensor, which need a lower refresh rate(10Hz or can be controled by **Sleep_freq** in table 14). The MSI does automatically switch to Active mode when finger is detected or by setting the POWER_MODE register to Active mode. Also, the MSI can automatically switch from Active to Sleep mode when no finger is detected for more than IDLE_PERIOD time, provided that ALLOW_SLEEP bit is set in the POWER_MODE register.Figure 47 shows how to force the slave into Sleep mode. Figure 48 Figure 45 shows how to force the slave to switch automatically into Sleep mode (set ALLOW_SLEEP bit in POWER_MODE register).

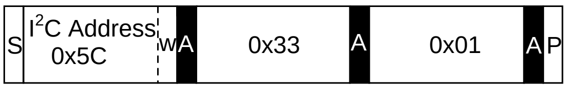


Figure 47: Sleep mode sequence

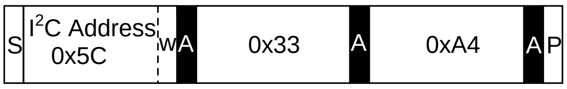


Figure 48: Sleep mode automatically switch sequence

2.4.3 Freeze mode

In this mode, the slave MCU internal clock source is stopped, and consumption is only MOS leakage.Figure 49 shows how to force the slave into Freeze mode.There are two ways to wake up from freeze mode.

- RST pin pull down (connect to the Ground) (default)
- ATTB pin change (“1 to 0” or “0 to 1”)

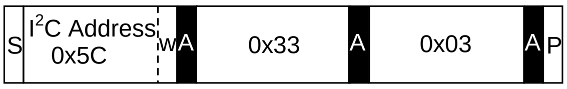


Figure 49: Freeze mode sequence

2.4.4 Power consumption

Table 22: Power Consumption list

Power Supply			3.3V			
Power Mode	Fingers	$F_{scan\ (Hz)}$	Min.	Typ.	Max.	Unit
Active mode	1	280		-	4	mA
	2	160		-	5	mA
	3	90		-	5.2	mA
	4	80		-	5.4	mA
	5	75		-	5.6	mA
Sleep mode	0	10		-	0.11	mA
Freeze mode	-			-	2	uA
bootload	-			-	6.2	mA
Calibration	-			-	6.2	mA

The figure 50 shows that how is the I^{2}C wake up in deep sleep mode.and it shows its Power Consumption .

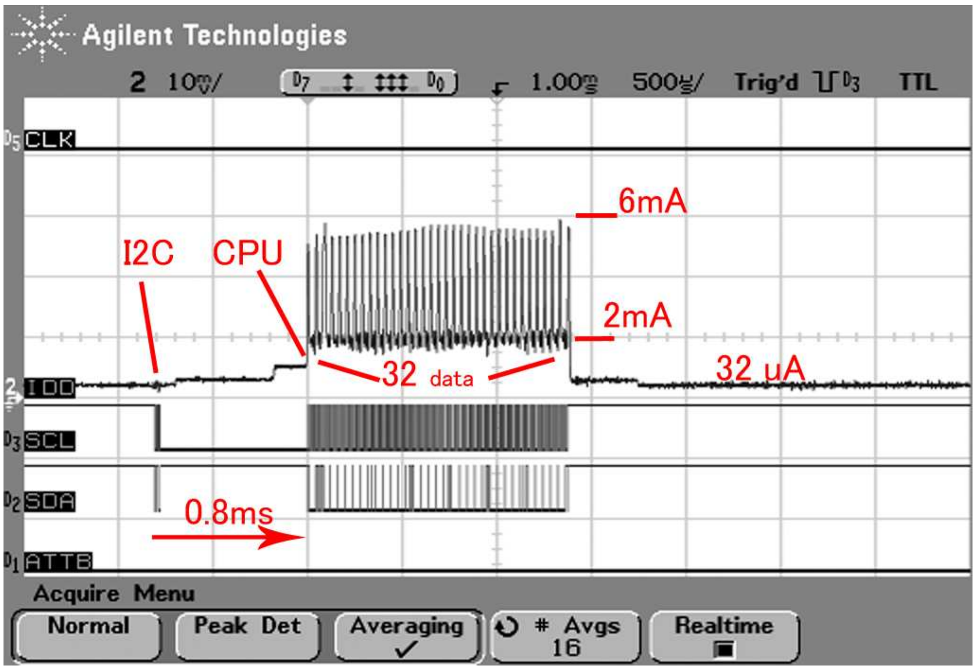


Figure 50: The example for I^{2}C wake up

2.4.5 Power mode diagram

Figure 51 shows the power mode change diagram.

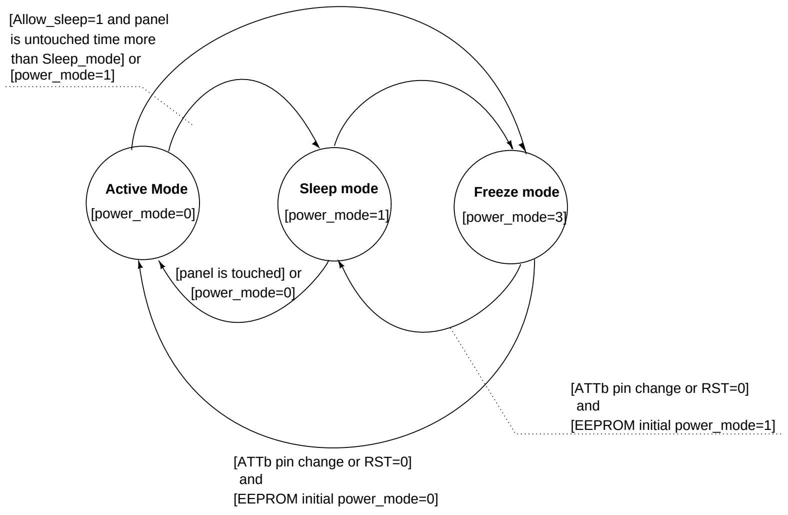

[Allow_sleep=1 and panel is untouched time more than Sleep_mode] or [power_mode=1]
Active Mode
[power_mode=0]
Sleep mode
[power_mode=1]
Freeze mode
[power_mode=3]
[panel is touched] or [power_mode=0]
[ATTb pin change or RST=0] and [EEPROM initial power_mode=1]
[ATTb pin change or RST=0] and [EEPROM initial power_mode=0]

Figure 51: Power mode diagram sequence

2.5 Special operations

The SPECOP register is used to request maintenance operations.

2.5.1 Normal Mode

When SPECOP is written with the code 0x00, the following read and write operation will transmit the RAM.

2.5.2 EEPROM read operation

EEPROM content can be read by the host. When SPECOP is written with the code 0x01, the following read operation will transmit the EEPROM memory.Master device will read the EEPROM memory from address previously stored in EEPROM_read_addr.

2.5.3 EEPROM write operation

The 512 bytes of the EEPROM can be written by the host through the MSI interface, as shown in figure 52. The SPECOP register must be set to 0x02. Following this command, ATTB line will be kept low during the whole operation. The MSI device now interprets write commands as addressed to the EEPROM. These write operation must only contain one byte of data, which means the requests contains EEPROM address (16 bits) and EEPROM data (8 bits). After each such writing request, a delay (no I^{2}C traffic) of $10\,\text{ms} > T_{\text{EEPROM}} > 4\,\text{ms}$ must be respected in order to guaranty the erase and write cycle of the EEPROM. The EEPROM writing mode is ended by performing a dummy read operation.

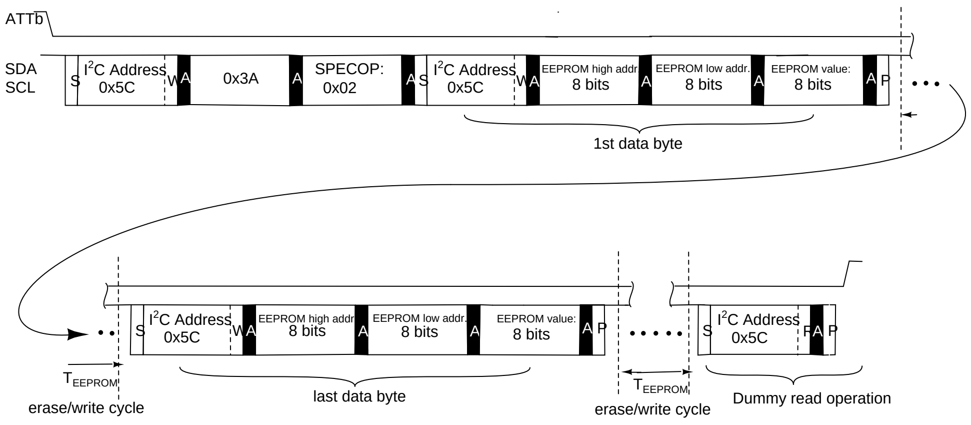


Figure 52: EEPROM write operation

2.5.4 Calibration

The objective of the calibration need to do one-time total calibration before product packaged and enable the border tracking calibration action in the product life.

One-time total calibration The objective of the one-time calibration is to measure the capacitance offsets of the electrode system when no finger is touching, and to store these values into EEPROM. After this, during normal operation, this calibration data is subtracted from the values measured. This operation should be necessary only once in the product life, as part of the final adjustment after the sensor module is mounted in its destination case. The calibration must be done with a clean sensor surface. When SPECOP is written with the code 0x03, the following will transmit to do calibration function and then he SPECOP register will take back its default zero value.The sequence for calibration is shown in the figure 53. Also the measurement for the calibration lasts no longer than 10 ms, the EEPROM erase and write cycle requires that the power supply be kept on for a duration of at most $T_{\text{CAL}} < 500$ ms (for one Tango solution). The ATTB line will be pulled up after finish calibration.

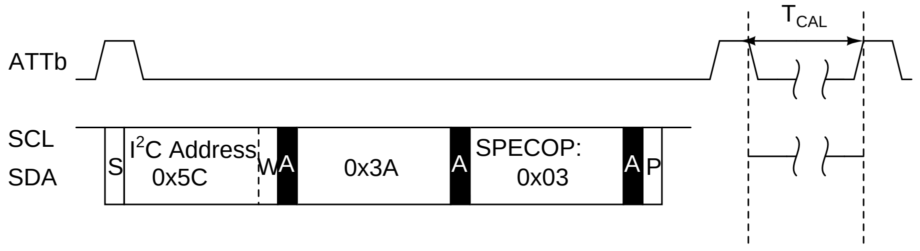


Figure 53: Calibration sequence

Border tracking calibration

Drift Analysis

Storage test (drift when panel is off; drift detected at startup):

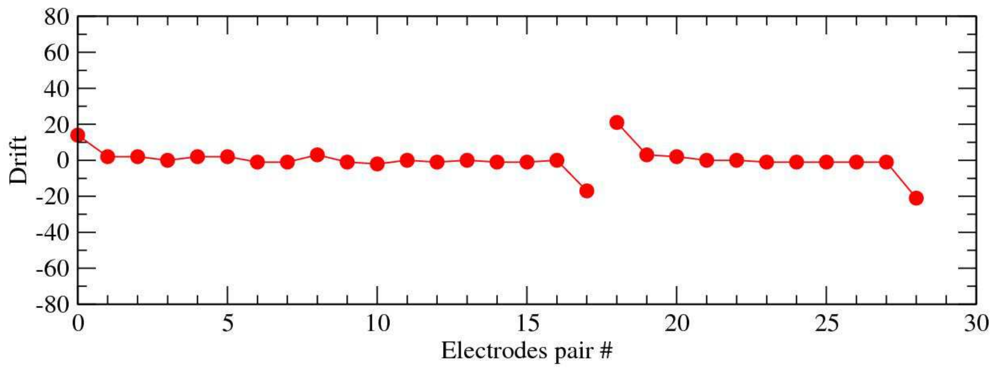


Figure 54: Storage test

High humidity test (drifting during the use):

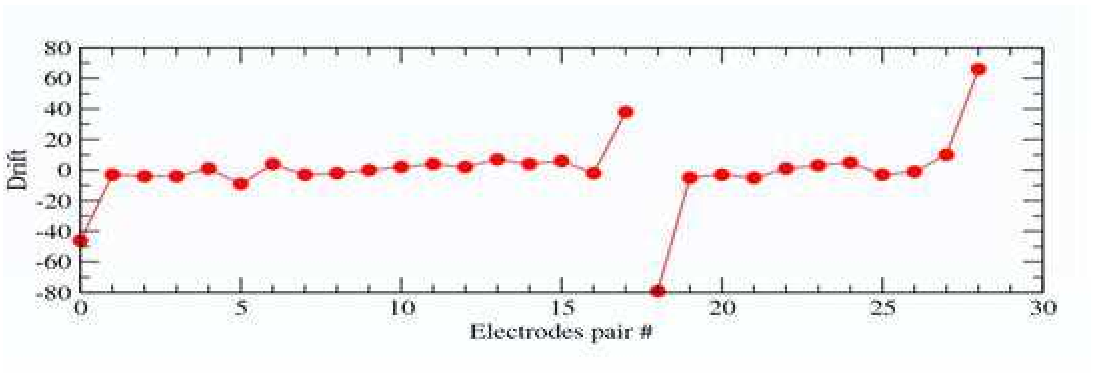


Figure 55: High humidity test

Conclusion: In both cases drift affects only edges.

Border drift value will be permanently monitored and adjusted. And drift change is much slower than finger signal change. Signal change speed:

- Few units/hour for drift signal
- Few unit/sec for finger signal

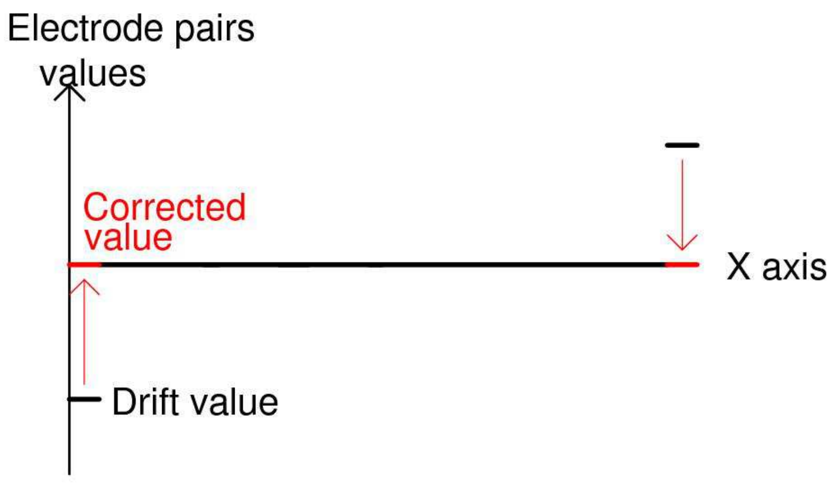


Figure 56: Border electrodes drift value

- Select “drift_comp” as 0x01 to active the border tracking calibration, please refer to chapter 9 EEPROM table.
- Border tracking algorithm permanently monitors and adjusts offsets of the 1st and last channels. The algorithm can differentiate cancel fingers from offset due to drift.
- Any of 1st channels has clipped, then there is no more visible drift, the system will fails.

2.5.5 CRC checksum

When `SPECOP` is written with the code 0x04, the following will transmit to CRC checksum operation, then the `SPECOP` register will take back its default zero value. The whole program memory checksum result will be wrote in the crc register.After each such writing request, a delay (no I^{2}C traffic) of $200\,\text{ms} > T_{\text{EEPROM}} > 150\,$ms(System Clock 8MHz) must be respected in order to guaranty checksum cycle of CRC.ATTB line will pull up when finish the crc checksum.zerocrossing

2.5.6 Bootloader

When `SPECOP` is written with the code 0x05, the following will transmit to bootloader operation, then jump to the start address of the bootloader section and the `SPECOP` register will take back its default zero value. See §section3

2.6 Coordinates characteristics

The reported posX and posY integer coordinates of the finger(s) position(s) are related to the indexes illustrated on figure 57, where X_0 to X_{n-1} and Y_0 to Y_{k-1} represent the X respectively Y electrodes. The scaling factor is 512 units per index pitch. For example, a finger touching the active area over X index 2.7 and Y index 1.4 will have the reported coordinates posX = 2.7 * 512 = 1382 and posY = 1.4 * 512 = 717.

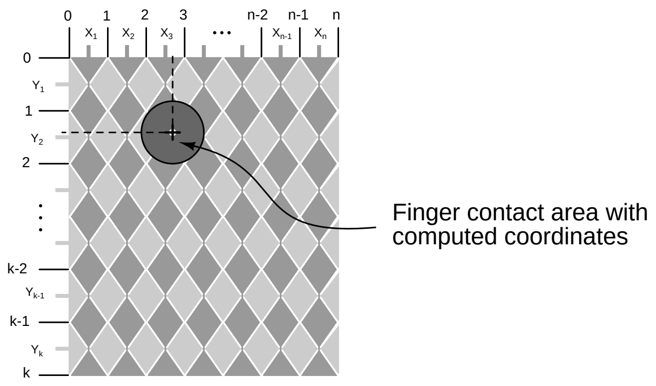


Figure 57: Active area coordinates indexes

As illustrated on figure 58, in the boarders of the active area the reported coordinate of the axis in which the finger is over or close to index 0 (X in this example) will be slightly shifted from the real finger position ($\triangle x \neq 0$ in this example), whereas the coordinate of the other axis (Y in this example) is not affected ($\triangle y = 0$ in this example). This is not any more the case in the accurate area which lays from X indexes 1.5 to (n-1.5) and Y indexes 1.5 to (k-1.5). Note that in any case, the coordinates are bounded inside the dark gray area shown on figure 59.

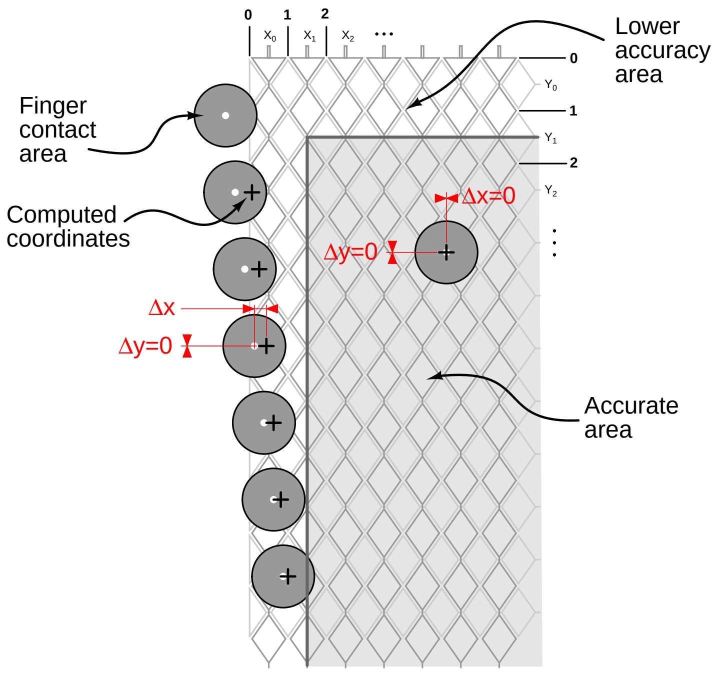


Figure 58: Accuracy of the coordinates vs real finger position on active area

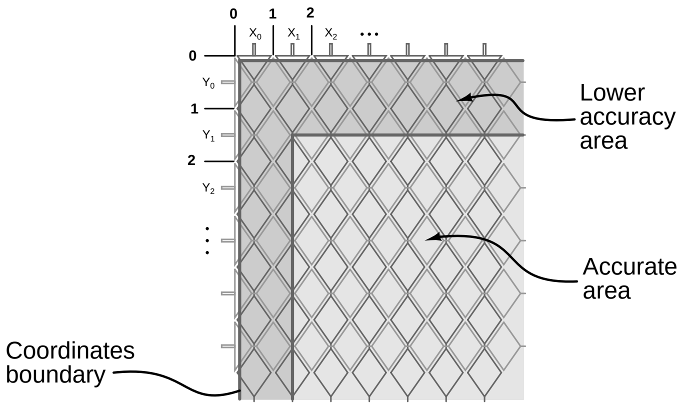


Figure 59: Active area accuracy zones and coordinates boundary

3 Bootloader

3.1 Introduction

The device can operate in two different modes:

Normal mode, which is the normal operation. The device uses the following I^{2}C address: 0x5C

Bootloader mode which allows loader operation. The device uses another I^{2}C address: 0x5D, and uses a specific syntax.

In bootloader mode, the device uses a different, more simple I^{2}C servicing firmware, which is totally separated from normal I^{2}C. This means that the entire firmware of normal mode can be reflashed, including the I^{2}C code itself

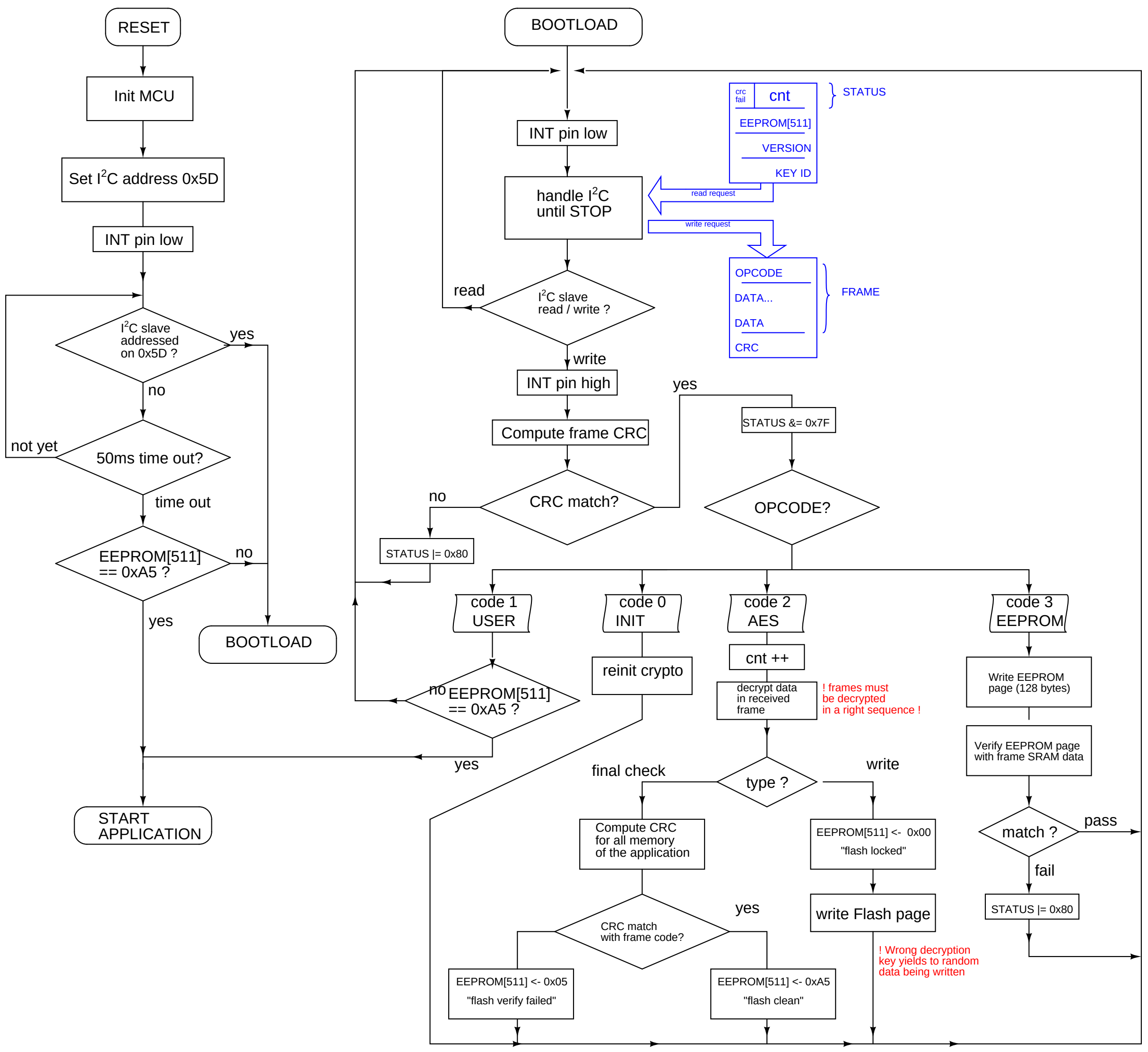


Figure 60: Bootloader flow diagram (for illustration purpose only)

3.2 Switching to bootloader

Switching from normal mode to bootloader mode can be done by two methods:

- by software, be modifying the SPECOP register by an I^{2}C command to address 0x5C.

- by hardware reset, followed by an appropriate I^2C command to address 0x5D.

3.3 Bootloader read operation

In bootloader mode, I^2C read operation always send the same sequence of information:

- STATUS reports a status register
- FLASHLOCK reports the flash status as stored in EEPROM[511]:
  - 0xFF means EEPROM is erased, flash status is unknown. Jump to application is impossible.
  - 0x00 means the flash has been locked. Jump to application is impossible.
  - 0xA5 means the flash has been unlocked. Jump to application is possible.
- VERSION reports the bootloader version
- KEY reports the customer cryptographic key (this is just a “label”, not the actual key!)
- Reserved: further read will issue a stream of zero. These positions are reserved for future use.

3.4 Bootloader write operation

In bootloader mode, I^2C write operation always follow the same sequence of data

- OPCODE, which is a command to execute, followed by optional DATA, a variable length sequence.

The OPCODE command is executed after the end of the I^2C transaction. The bootloader can understand the following OPCODE commands:

- INIT (code 0): reinitialize the cryptography engine, and allow for a new upgrade
- USER (code 1): switch to normal mode.
- AES (code 2): load a frame. Frames are decrypted and interpreted after the I^2C STOP transition. The device will release the INT signal and will not respond to any I^2C solicitations during the execution of this instruction. These pages gave a secondary OPCODE, which selects either:
  - Actual writing of FLASH pages
  - Final check of the whole FLASH and unlocking
- EEPROM page (code 3): allow user to modify EEPROM without cryptography. The CRC error flag is set if either transmission or writing to the EEPROM failed

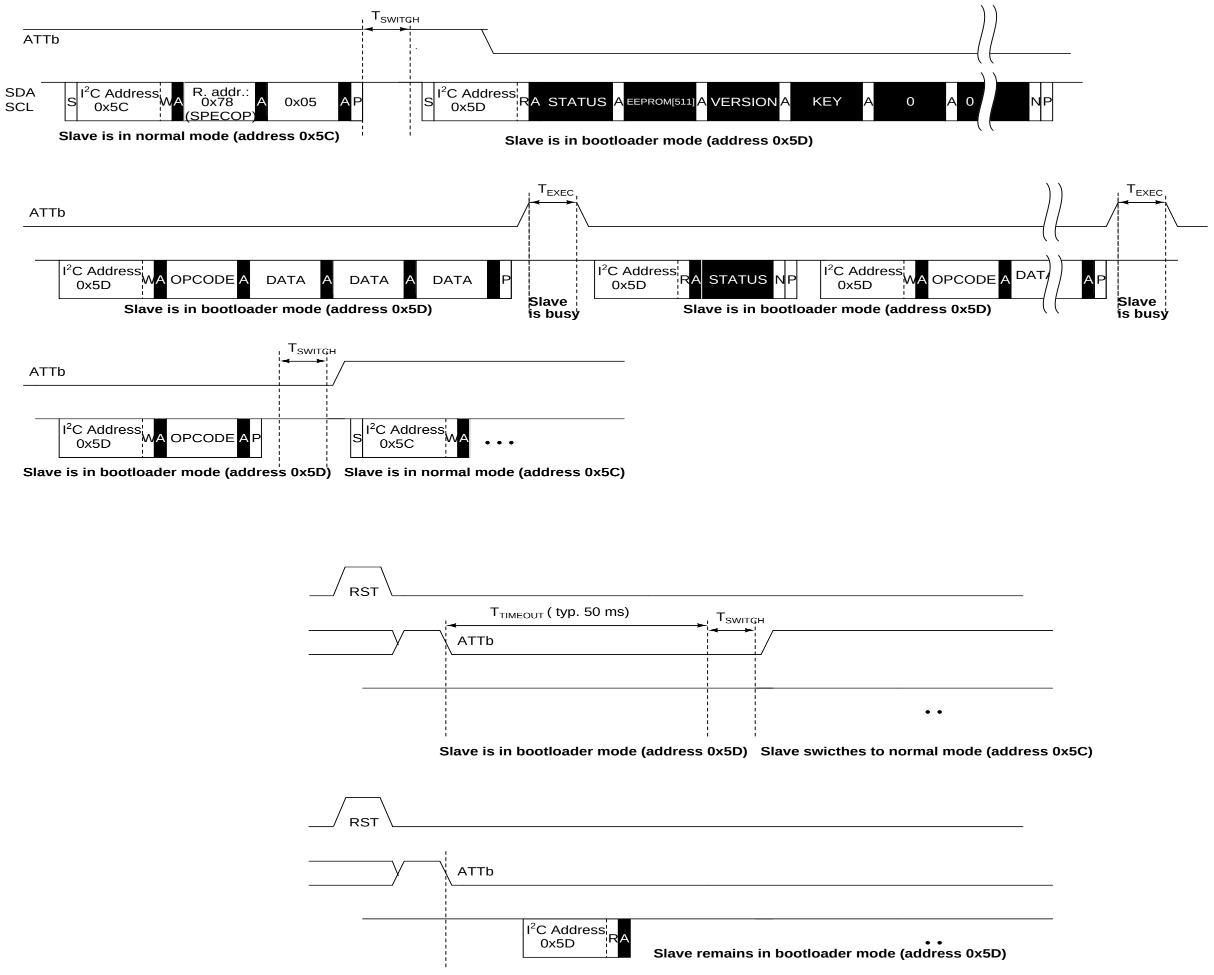


Figure 61: Examples of I^{2}C communication. Top row shows the transition from normal to bootloader mode, followed by a read operation which reads the status, bootloader version and crypto key information. The row in the middle shows the sequence of sending data frames, followed by a flashing operation, follow by checking the status register and a continuation of frame loading. The bottom row shows the switching from bootloader back to normal mode.

3.5 File format

3.5.1 Header

The header contains two lines starting with # sign:

- First line is the bootloader version (in hex code, single byte)
- Second line is the key identifier (in hex code, single byte)

3.5.2 Frame format

The frame format is shown in figure 62.

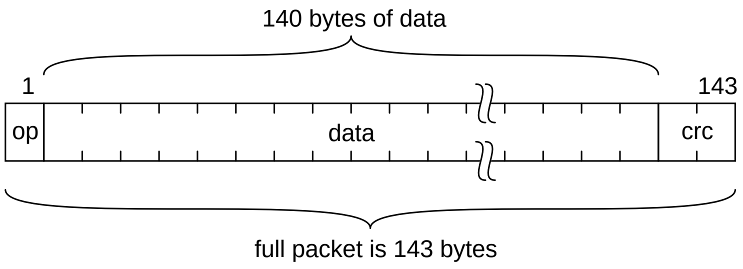


Figure 62: Format of a encrypted frame

The OPCODE (1 byte, code 02) and the CRC (2 bytes) values are already prepared and are already contained in the ASCII file. The format of the ASCII file is shown here under. Each frame starts with a '$' sign and ends with a UNIX like return carriage '\n'. Each frame is 143 bytes long, coded in HEX.

3.5.3 File example

```
#01
#A0
! Comment example
$023FD38AE5706C42BCDFD533DFD39D860C90FF9672AFC69E47CA035BE2775F
$029C081C92AD4FC9E74B000D0B669B0445E86DFC37824D0D7702D4BC6DBC34
! Comment again here
$02E3C7B35082AD22A61DF0EB571F758DFB61ED6D27964EE99B44AD5D31317D
```

The CRC of the part is computed out of the 141 first data bytes (it therefore includes the OPCODE field). The CRC function is CRC-16 with polynomial commonly called 0x8005. It is optional for the host bootloader to check the CRC value (this only prevents data corruption of the ASCII file).

3.6 Error handling

Table 23: Status register

Bit	Symbol	Description
7	CRCERR	Set to 1 if the computed CRC does not match the transmitted CRC.
6	FRMCNT6	Frame counter. This counter counts the number of successfully received encrypted frames. Should a CRC error occure, the host should send the frame following the FRMCNT value. For example, if a CRC error occures and if FRMCNT==3, the host should send againt the fram #4. Neither the frame counter nor the CRCERR flags are indication of correctness of the decryption key.
5	FRMCNT5	
4	FRMCNT4	
3	FRMCNT3	
2	FRMCNT2	
1	FRMCNT1	
0	FRMCNT0	LSB

3.7 Example

The figures 63 to 66 show a typical sequence of firmware upgrade. The total time is 7 seconds.

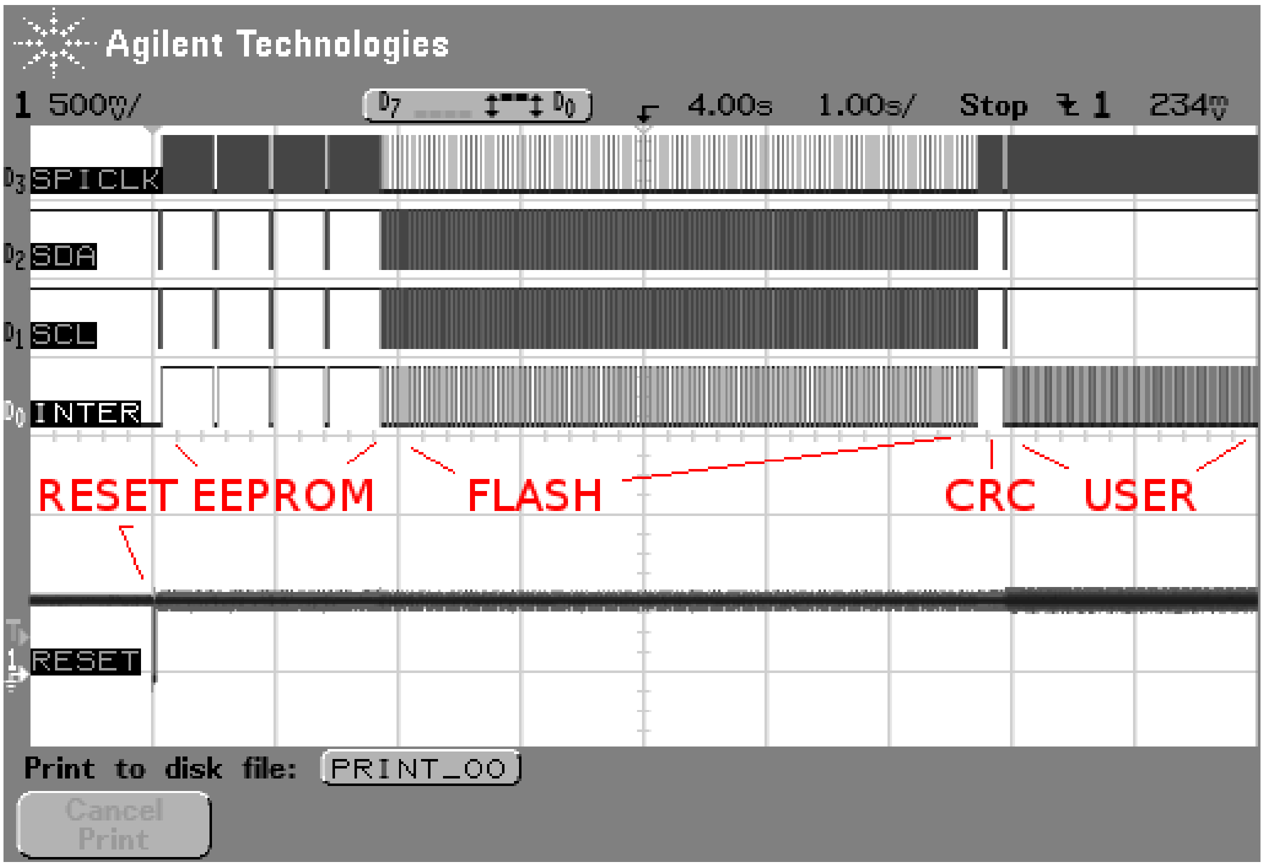


Figure 63: Overview of a full firmware upgrade. Reset pulse is followed by 4 EEPROM pages, then by 112 FLASH pages, then followed by a full CRC of the flash, finally, the system jumps into user application mode and start of scan, recognized by the high activity of SPI clock.

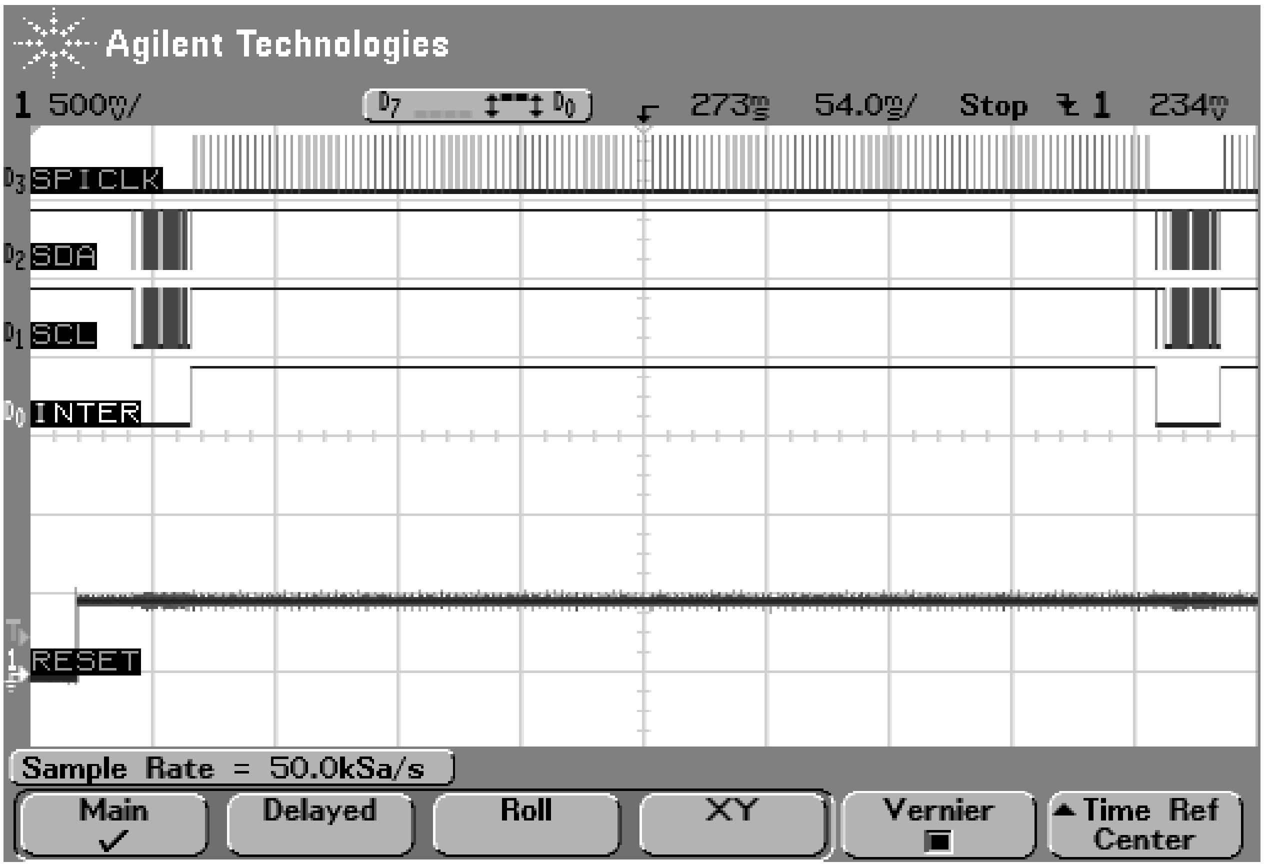


Figure 64: Detail of a EEPROM page, where SPI clock shows 128 pulses, one for each byte.

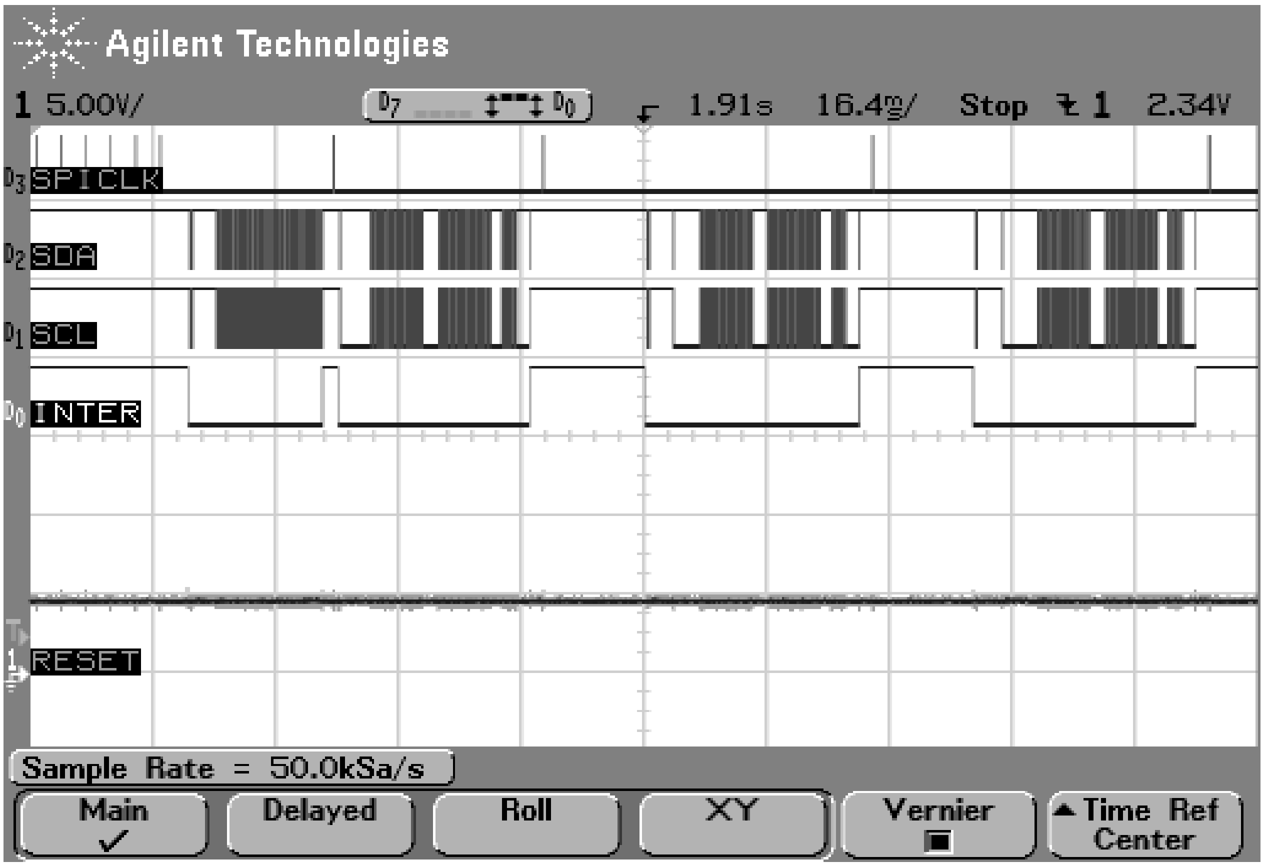


Figure 65: End of EEPROM pages, followed by a reset of the cryptography machine, and followed by some some FLASH pages.

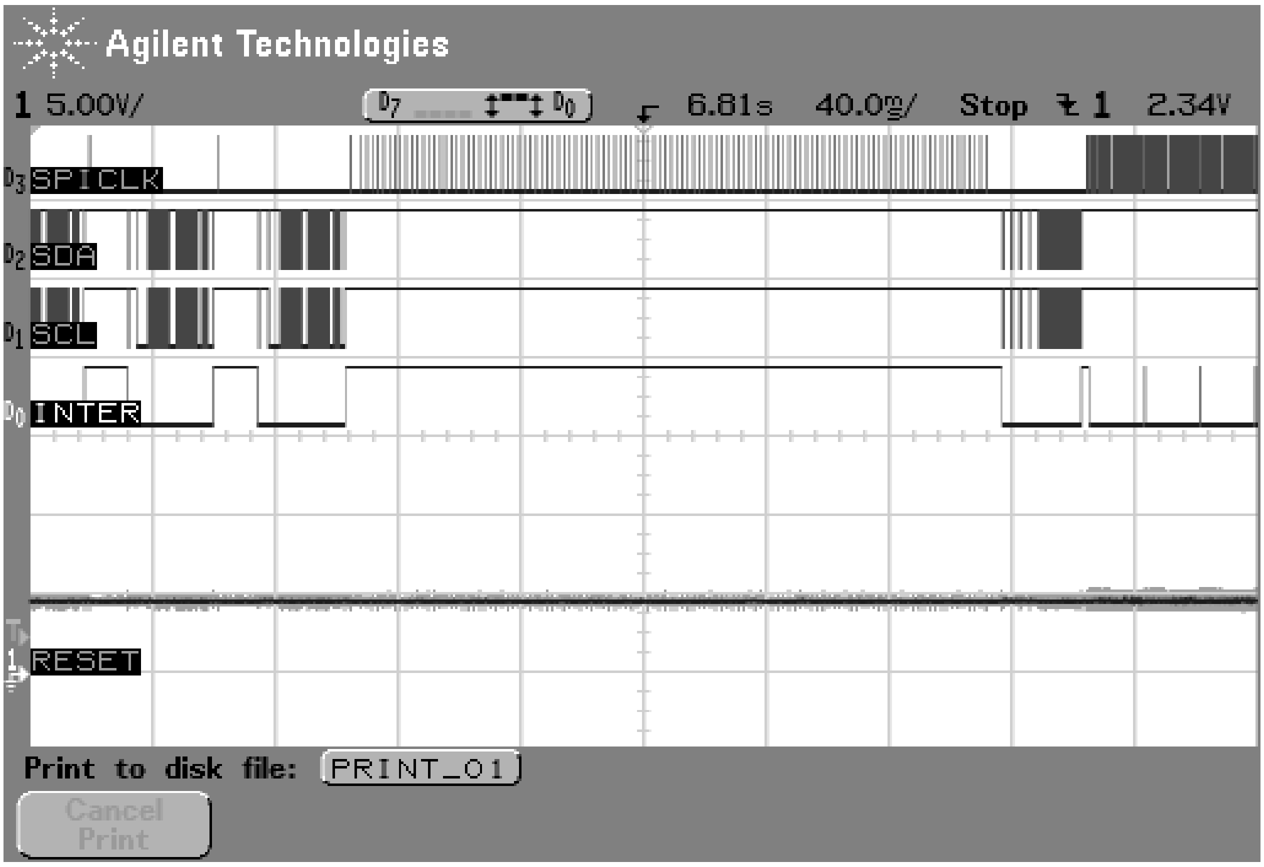


Figure 66: End of FLASH pages writing, followed by a check of the full 112 pages of FLASH, visible as 112 pulses. Then the system jumps into normal application, which shows a high activity of SPI clock.

4 EEPROM parameter

4.1 EEPROM features

Tango Core eeprom stores all the project related parameters for use with the firmware. These parameters include pinmap, scanning patters, SPI speed etc.

The parameter structure within EEPROM is defined in the firmware as c structure: eeprom_t.

4.2 Structure members

Algorithm 2 Top level parameter structure

```
typedef struct
{ uint8_t version[32];
eep_hal_t hal;
eep_sal_t sal;
eep_salp_t salp;
eep_algo_t algo;
}
eeprom_t;
```

There are five members in eeprom_t:

version 32 character string storage reserved for customer use

eep_hal_t Hardware abstraction layer, a structure by itself which contains sub members

eep_sal_t Setup abstraction layer, a structure by itself which contains sub members

eep_salp_t Setup abstraction layer plus, a structure by itself which contains sub members

eep_algo_t Algorithm layer, a structure by itself which contains sub members

4.3 version

4.3.1 version

32 bytes are allocated for customer's use. The content does not influence the operation of the firmware. The content can be retrieved by .....

4.4 eep_hal_t hal

Hardware Abstraction Layer (HAL)

This part auto generate from **xml** tool, customer can't modified them.

4.4.1 EEP_CUSTOMER_NAME

Put customer company name, max 11 characters. The content does not influence the operation of the firmware, just for user mark.

4.4.2 PRODUCT_INCH

Recorded touch screen size, the content does not influence the operation of the firmware, just for user mark.

4.4.3 EEP_RES_X

Resolution of X asix.

4.4.4 EEP_RES_Y

Resolution of Y asix.

4.5 eep_sal_t sal

Setup Abstraction Layer (SAL)

4.5.1 EEP_X0N

Number of electrodes for X axis.

4.5.2 EEP_X0_OPTION

bit	function
0	bit 0-2 SPI channel set
1	0b000 SPI 0, 0b001 SPI 1
2	0b010 SPI 2
3	unused
4	unused
5	unused
6	Tango chip type: 0:32 scan pins, 1:48 scan pins
7	unused

Table 24: EEP_X0_OPTION

4.5.3 EEP_X1N

Redundant

4.5.4 EEP_X1_OPTION

Redundant

4.5.5 EEP_YN

Number of electrodes for Y axis.

4.5.6 EEP_Y_OPTION

Same function as EEP_X0_OPTION.

4.5.7 EEP_SPI_SPEED_ME0_X

Lift time for METHODE 0 when scanning X axis. Tango Core MCU frequence is 24Mhz, if you need the lift time T$_{lift\,up}$ is 1μs, set value is 24 (42ns·24=1μs), also reset time T$_{lift\,down}$ be changed in same.

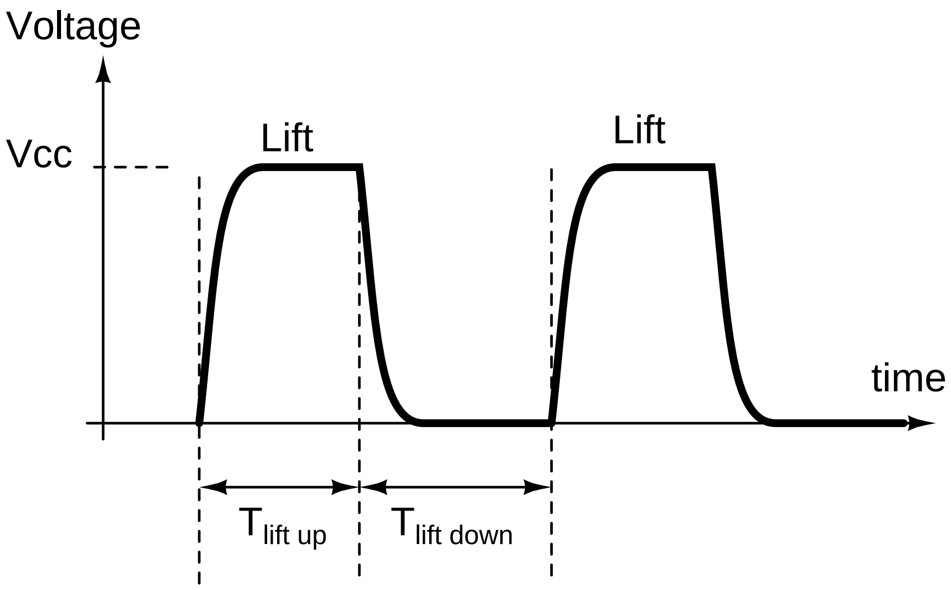


Figure 67: Lift time

4.5.8 EEP_SPI_SPEED_ME0_Y

Lift time for METHODE 0 when scanning Y axis.

4.5.9 EEP_SPI_SPEED_ME1_LX

Lift time for METHODE 1 when scanning X axis.

4.5.10 EEP_SPI_SPEED_ME1_LY

Lift time for METHODE 1 when scanning Y axis.

4.5.11 EEP_M1_MEDIAN_CONV

Number for measurement times in the M1 conversion. The measured values are sorted and the median value is selected as a result. Value range is 1-9, default is 9.

4.5.12 EEP_M1_MEDIAN_CMP

Number for measurement times in the M1 comparison. The measured values are sorted and the median value is selected as a result. Value minimum is 1, maximun 255.

4.5.13 EEP_M0_FAST_MODE

Specifies the behaviour of repeat measurement as shown below. The value of 2 will set the firmware to auto mode which uses the threshold specified in EEP_M0_AUTOFAST_THR. See below for more description. Set this value will control M0 measurement time. If value is 2, firmware will measure once, but if this time measurement result with large difference firmware will measure twice again, use all 3 times results to get a median result.

value	function
0	5 times
1	3 times
2	auto or 3
3	1 time

Table 25: M0 fast mode

4.5.14 EEP_M0_AUTOFAST_THR

This is a threshold value only to be used when auto mode is enabled (EEP_M0_FAST_MODE = 2). If the difference between the current and the previous measurement is above this threshold, then the firmware performs two additional measurement (3 measurements in total). Therefore, if you set this value lower, the firmware will perform slower but the result will be less noisy. On the other hand, if this value is higher, the firmware will perform faster but the result will have more noise. Default value is set to 15.

4.5.15 EEP_DICHO_ACCURACY

Resolution of the dichotomy measure: 0 => 8 bits, 1=> 7 bits, 2=> 6 bits, 4=> 5 bits, 8 => 4 bits, etc...

value	function
0	8bits
1	7bits
2	6bits
4	5bits
8	4bits
...	...

Table 26: M1 ACCURACY

4.5.16 EEP_DICHO_MEDIANE

This parameter specifies the number of conversion performed on each bit on dichotomy. Dichotomy is different from the conversion. Conversion does compare each bit only once, but with dichotomy the conversion is performed repeatedly for specified time and the median is used as result .Value should be odd number :1 3 5 7 9. Usually is 5.

4.6 eep_salp_t salp

4.6.1 EEP_M0_RAW_LOWPASS

Enable this function make Raw data more stable. Set value is 0 unable, 1 enable.

4.7 eep_algo_t algo

Algorithm layer(ALGO)

4.7.1 EEP_DISTO_HEAVY_ME0

Varying finger pressure often causes the linearity error. As shown the figure below, dotted line is actual finger move and solid line is what the firmware see as finger movement. This kind of linearity errors can be corrected by a transfer function.

EEP_DISTO_HEAVY_ME0 is a 32 bytes table for high finger pressure linearity calibration. This parameter is used in the transfer function. Please see separate document **Application note 510** for more information.

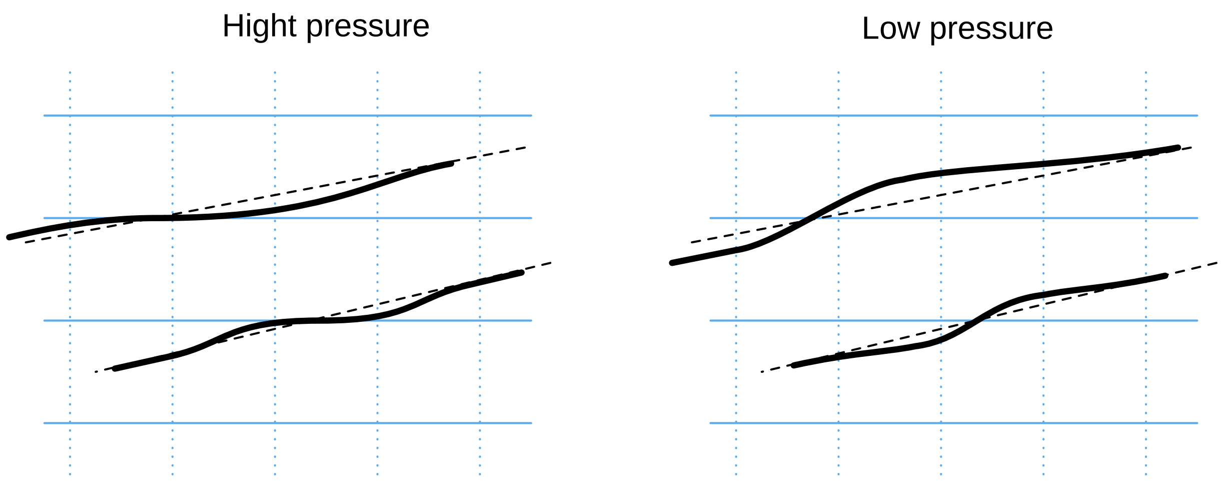


Figure 68: Finger Pressure cause linearity error

4.7.2 EEP_DISTO_LIGHT_ME0

It’s a 32 bytes table for light finger pressure linearity calibration.

4.7.3 EEP_BORDER_GAIN_DISTO

This parameter extends the data points appear when the finger is closer to or on the borders. The range is between 0 and 255. As shown in figures below, if value is too low, the drawing cannot follow the finger which is moved close to the border. If the value is too high, too many extended points will appear on the drawing.

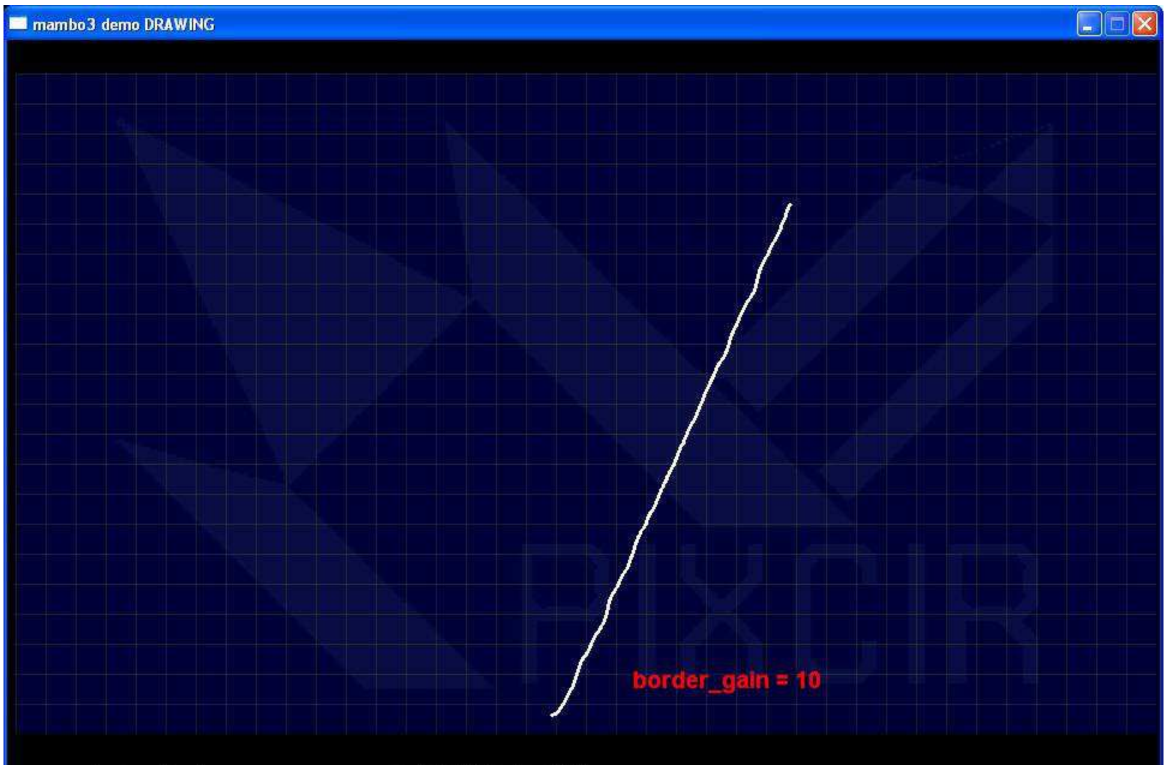


Figure 69: Drawing does not follow the finger movement

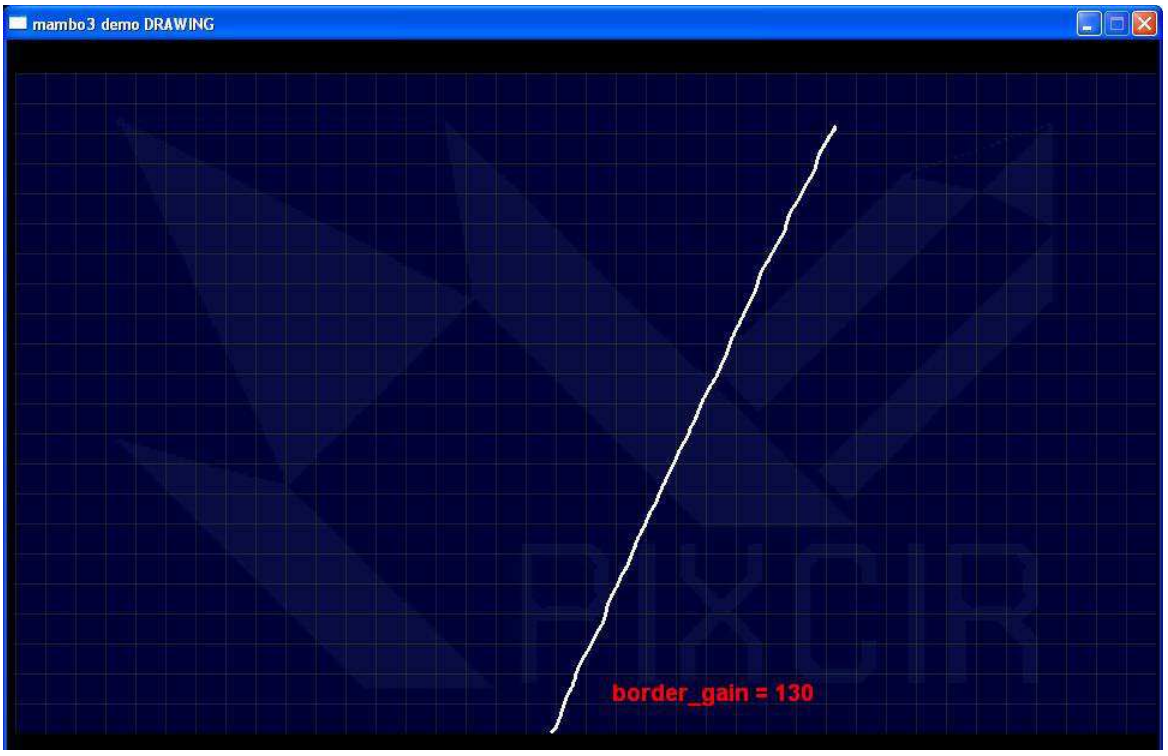


Figure 70: Drawing follows the finger movement correctly

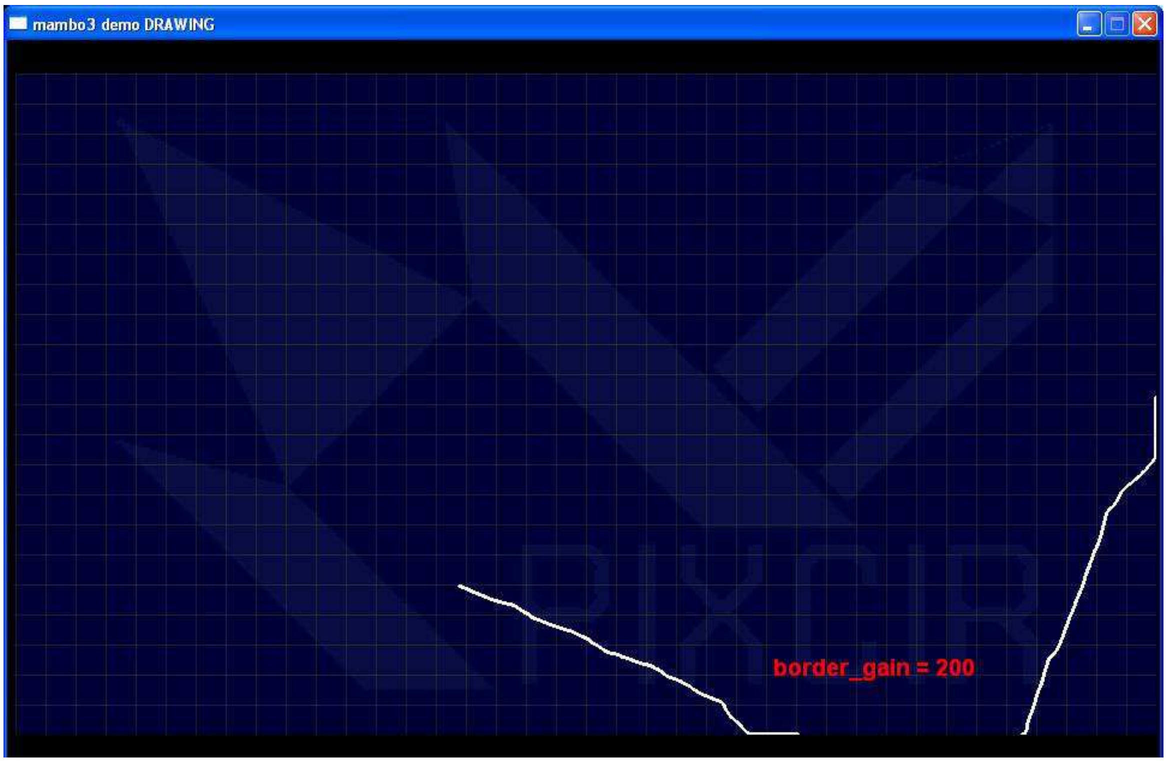


Figure 71: The too many extended points appeared on the drawing

4.7.4 EEP_DRIFT_COMP

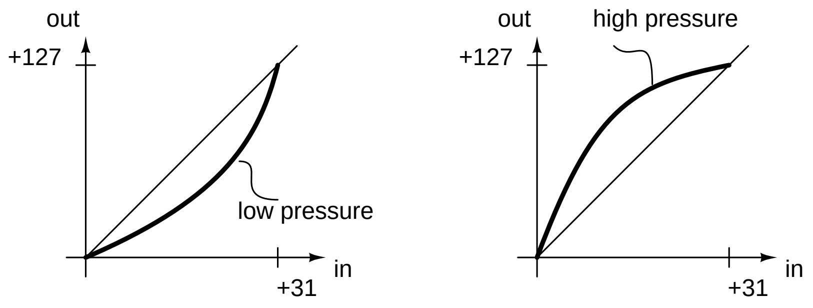
Please see section 2.5.2 drift. Set value is 1 enable drift compensation, 0 unable it.

4.7.5 EEP_AMINX

Parameter for X axis.

When the finger touches the panel, the pressure level is measured with the amplitude of RAW data, For example we describe a pressure include 60% high pressure and 40% low pressure. Then we use 60% high pressure corrections curve and 40% low pressure corrections curve.

Look at figure 72we can see two parameters **Amin** and **Amax**. At left of **Amin** is 100% light pressure, at right of **Amax** is 100% high pressure in the middle is 50% high and 50% high pressure. Modify these two parameters can control gradient and force pressure into high or low pressure region.



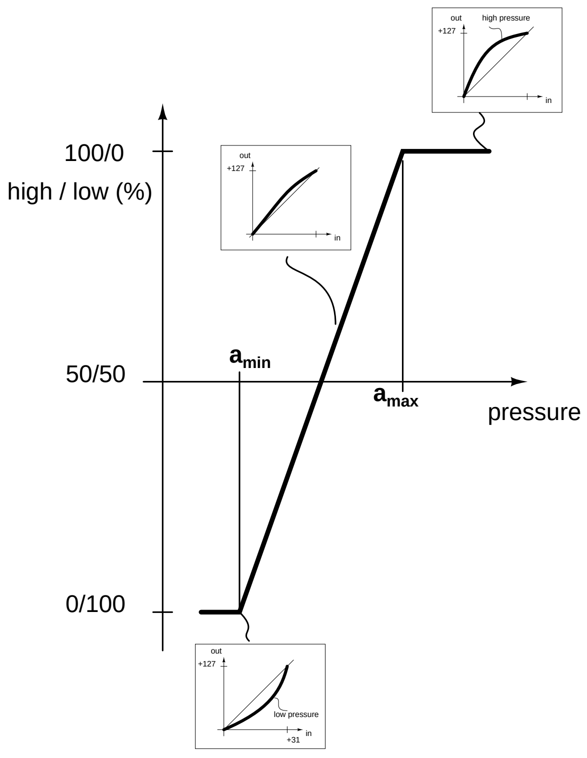


Figure 72: Pressure coordinate system

4.7.6 EEP_AMAXX

Amax parameter for X axis.

4.7.7 EEP_AMINY

Amin parameter for Y axis.

4.7.8 EEP_AMAXY

Amax parameter for Y axis.

4.7.9 EEP_M0_X_THR_ONEFINGER

One finger threshold for X axis. When we move our finger to draw a line on the panel, the distance between the finger and panel will change. This distance corresponds to the signal amplitude sensed from the panel (see the difference of Raw data in figure). Firmware will calculate Raw date get the signal level: **signal X** (see Table 15: INT_mode register: signal_x) is above this threshold value (i.e. finger is closer to the panel), the firmware recognize the signal as a finger. If not (i.e. finger is too far from the panel), the firmware does not recognize the signal as a finger. If the threshold set close to signal X (above limit) will cause “ghost” recognize what we don’t need touch, in another hand set threshold close to signal X (below limit) will cause break line. So modify threshold value in the middle area between the two kinds signal shown in figure 73. Default value is 40.

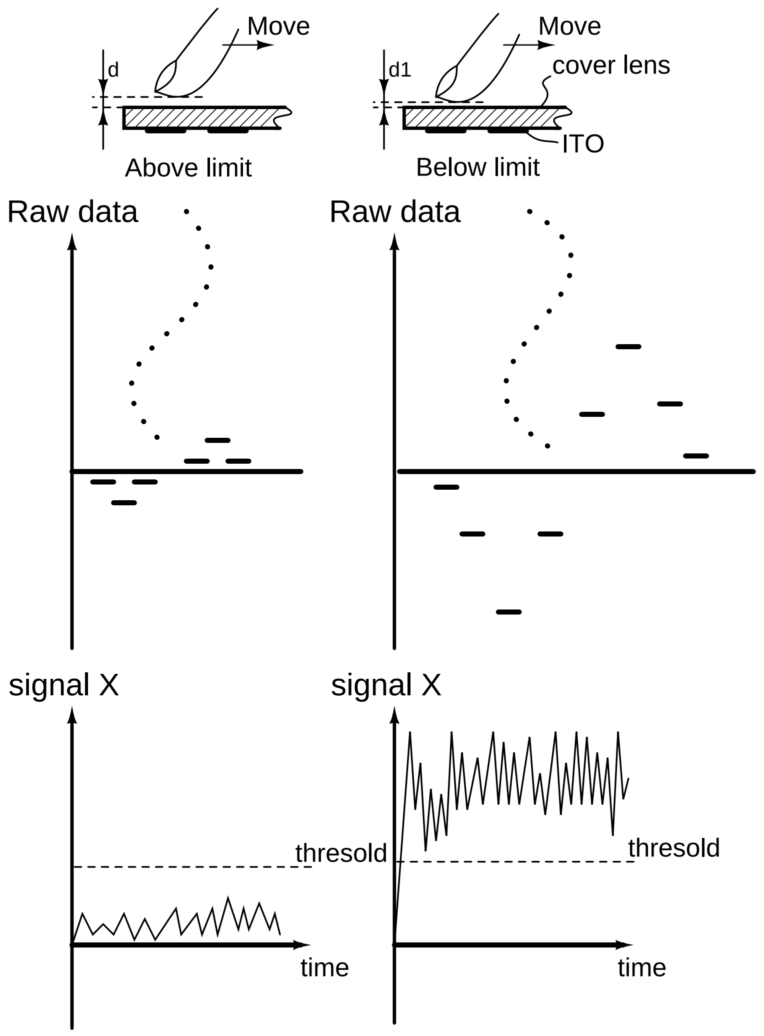


Figure 73: Finger distance to panel below or above the limit

4.7.10 EEP_M0_Y_THR_ONEFINGER

One finger threshold for Y axis.

4.7.11 EEP_M0_THR_ZEROCROSSING

A finger touch or noise both can make raw data active (i.e. recognize as a finger). The following 2 conditions have to be met in order for the signal to be recognize as a finger by the firmware. The signal: 1st raw data value from negative to positive, crosses the zero line and 2nd the difference between the last negative raw data to the first positive raw data must large than zerocrossing value. Default value is 20.

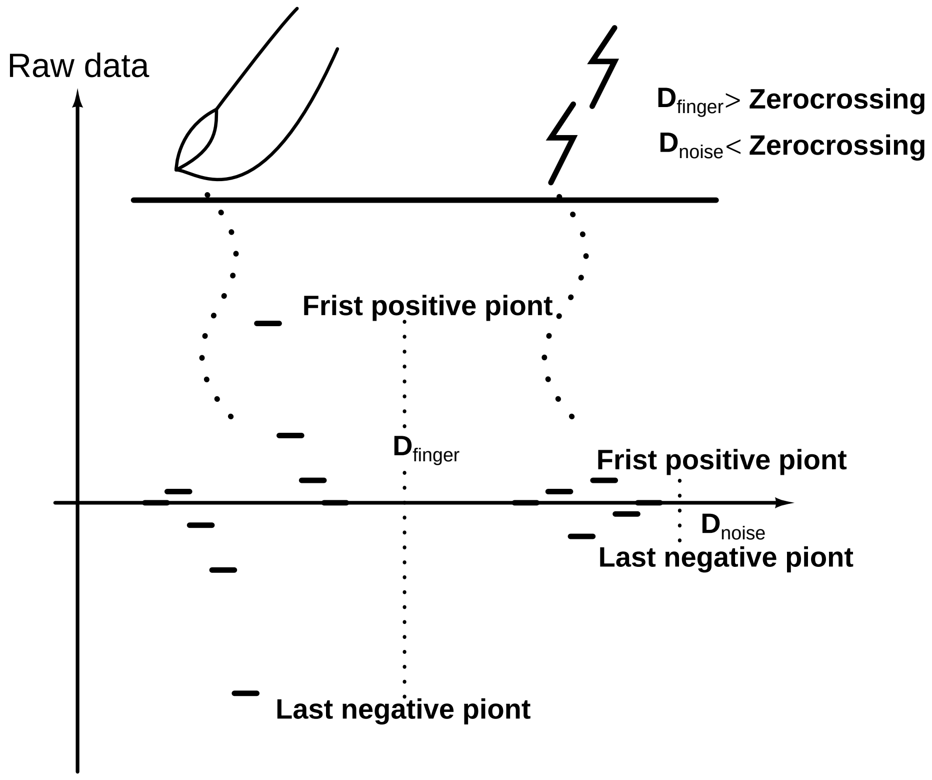


Figure 74: zerocrossing

4.7.12 EEP_M0_X_THR_MULTIFINGER

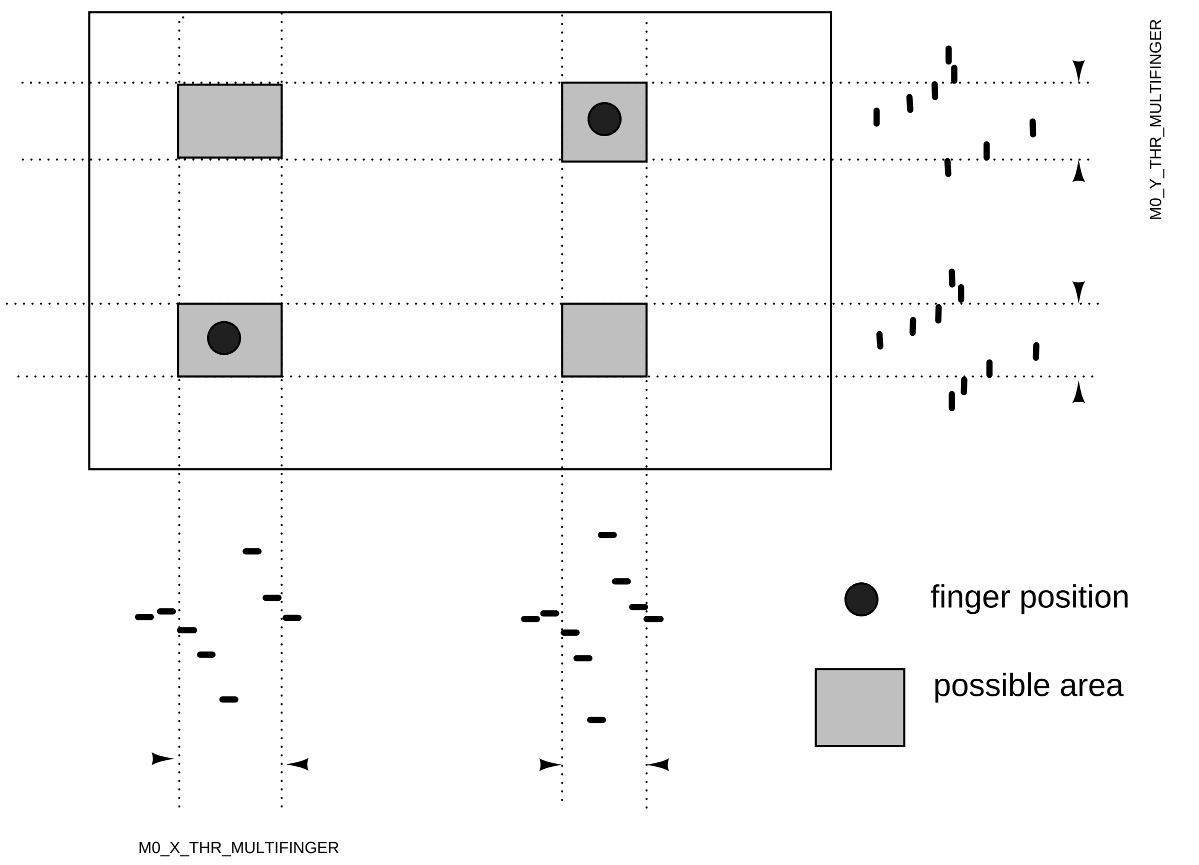


Figure 75: M0_multifinger_thr

Multfinger threshold for X axis. Firmware detects raw data to find out the possible area which will be further scan(M1). Modify this threshold will change the size of possible area. Over large value will cause scanning speed slow down, too small value will increase risk of lose finger position.

Normally this threshold less than the one for single finger (EEP_M0_THR_ZEROCROSSING). Default value is 20.

4.7.13 EEP_M0_Y_THR_MULTIFINGER

Multfinger threshold for Y axis. Default value is 20.

4.7.14 EEP_M1_thr_lx_multifingers

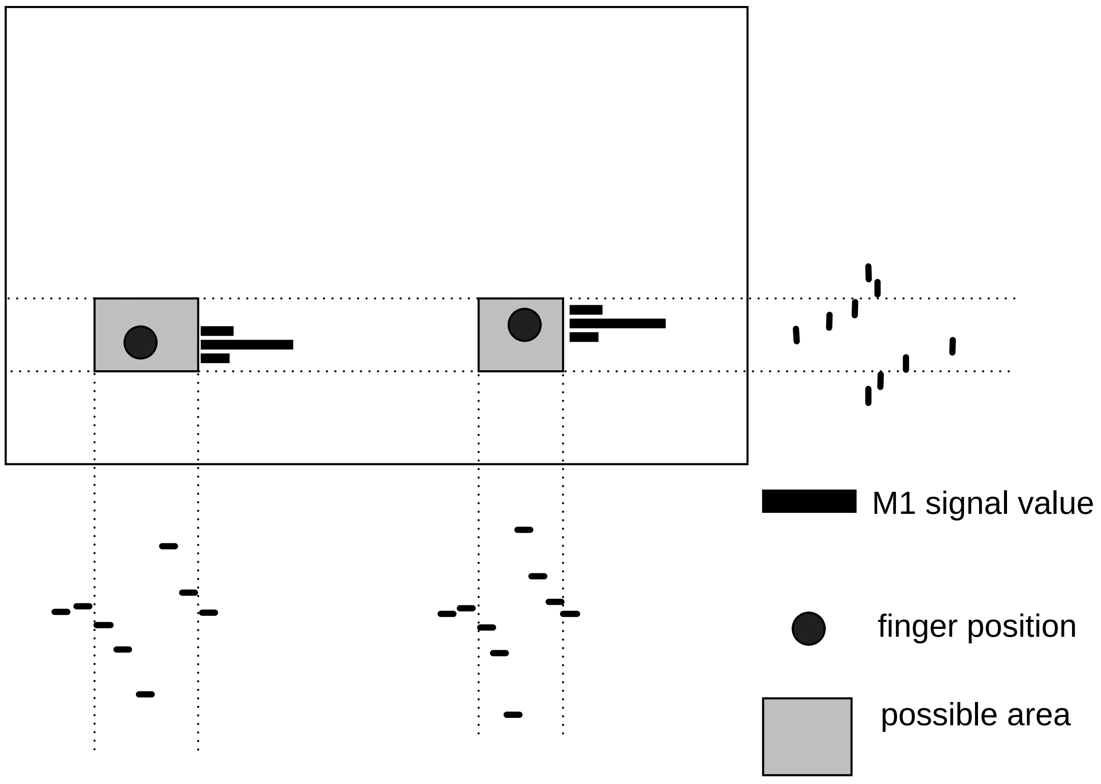


Figure 76: M1_thr_lx_multifingers

Multifinger threshold for whole panel. If put multifinger in one line (similar X axis coordinates or Y axis coordinates). It's hardly detected accurate each finger position by raw data signal. Firmware will do 3 more M1 scan nearly the possible finger position . If the largest value (sways in middle) larger than this threshold, finger position be detected. Default value is 30.

4.7.15 EEP_M1_thr_ly_multifingers

Multifinger threshold for whole panel. If M1 signal stronger than this threshold we confirm that this area is a really finger position. Default value is 10.

4.7.16 EEP_POWER_MODE

Same as section 2.2.4Power_mode register. Default value is 0x04 (working in active mode and enable Allow sleep bit).

4.7.17 EEP_INT_MODE

Please check section 2.2.5 INT_mode register.

4.7.18 EEP_INT_WIDTH

Only use for INT_mode choose ATTB is pluse. Modify this value to change each pluse width. Such as value is 200, pluse width is 1ms.

4.7.19 EEP_MOVE_THR

Only use for INT_mode (bit 1 and bit 0 in INT_mode register) set to "01". This parameter is related with resolution. For example one panel resolution are X:1024,Y:768 and EEP_MOVE_THR is 20, only finger move out of 20X20 square, will send ATTB pluse and update finger postion. **Attention** :If this value set too large will cause finger type unease be detected.

4.7.20 EEP_SLEEP_DLY

The delay time, the start is the last touch released in Active mode and the end is switch into Sleep mode succesful.

4.7.21 EEP_SLEEP_FREQ

Scanning frequence in Sleep mode.

4.7.22 EEP_FINGER_SEP_MODE

Depends on the physical characteristics of each kinds of touch senor; different size of finger will cause finger detected issue. Such as in some case very big thumb be recognized as two very close fingers; and sometime female small finger or finger nail drawing will not be detected. In panel tunning step we need to see the finger drawing performance to choose which value is the most suitable for panel.

value	function
0	Automatically distinguish finger size
1	small finger
2	normal finger (default)
3	huge finger

Table 27: FINGER_SEP

4.7.23 EEP_ATTB_REDUCTION

Reduce report rate for slow host. Such as value "4", Report rate will reduce as 1/4 as normal.

4.7.24 EEP_IO_SWITCHING_LEVEL

Switch the voltage of the interface IO (SDA, SCL, ATTb, AUX). If 0 => Vdd/2, if 1 => 0.9V. The current bootloader v47 is configured to use Vdd/2 as switching level.

4.7.25 EEP_BUTTON_SCAN

button scan bit; enable=1, disable=0.

4.7.26 EEP_BN

Number of buttons to scan.

4.7.27 EEP_BUTTON_USE_INTEGRAL

Use integral of each raw values for multiple button touch detection 1=On,0=Off.
a. Single touch: EEP_BUTTON_USE_INTEGRAL=0. All the buttons are explicitely scanned and button touch detection is applied on each individual raw signal from button electrode. This does NOT work in the case when multiple touch is occured.
b. Multi touch: EEP_BUTTON_USE_INTEGRAL=1. The last EEP_BN'th button is not scanned (forced to zero) and each raw values are integrated one by one (from EEP_BN'th button to 1st button) so that we can detect multiple touch upto EEP_BN-1 buttons.

4.7.28 EEP_SPI_SPEED_ME0_B

SPI speed specific for button scanning. See EEP_SPI_SPEED_ME0_X for more information.

4.7.29 EEP_BUTTON_LOOP_COUNT_MAX

All the buttons are scanned once in every this max scanning loop (of X/Y axis) count. It means smaller value will faster refresh buttons' rawdata.

4.7.30 EEP_BUTTON_NOISE_THR

If the raw data contains the noise of which level is between -4 to +4, this can be removed (raw = 0) with this threshold value of 5.

4.7.31 EEP_BUTTON_RAW_MAX_CLIP

Use under mult-touch function enable. The raw signal is clipped to this value before integral (pos & neg). Setting this value lower than default 255 may help to achieve consistent response from the buttons. However, the effectiveness of this parameter may vary and depends on various factors such as electrode layout etc.

4.7.32 EEP_BUTTON_DETECT_RISING

Raw signal polarity: 1=rising, 0=falling edge detection.

4.7.33 EEP_BUTTON_DEBUG_DISPLAY

Set 1 show button raw value instead of Y axis by AUX pin on Mambo (Pixcir's graphical tools) Debug mode. Set 0 the button raw values appear in separate section starting 'B' next to Y axis, can be seen by hyper terminal in text.

4.7.34 EEP_BUTTON_THR_PRESS

Threshold for button press. When the button is not pressed in previous scan and the button raw value becomes above this threshold, then the button flag (in I2C table) is set to 1 (pressed). The actual threshold value used is double this value. For example, if you set this as 50, the actual threshold is 100. After calibration then choose each button's threshold.

4.7.35 EEP_BUTTON_THR_RELEASE

Threshold for button release. Reference above PRESS.

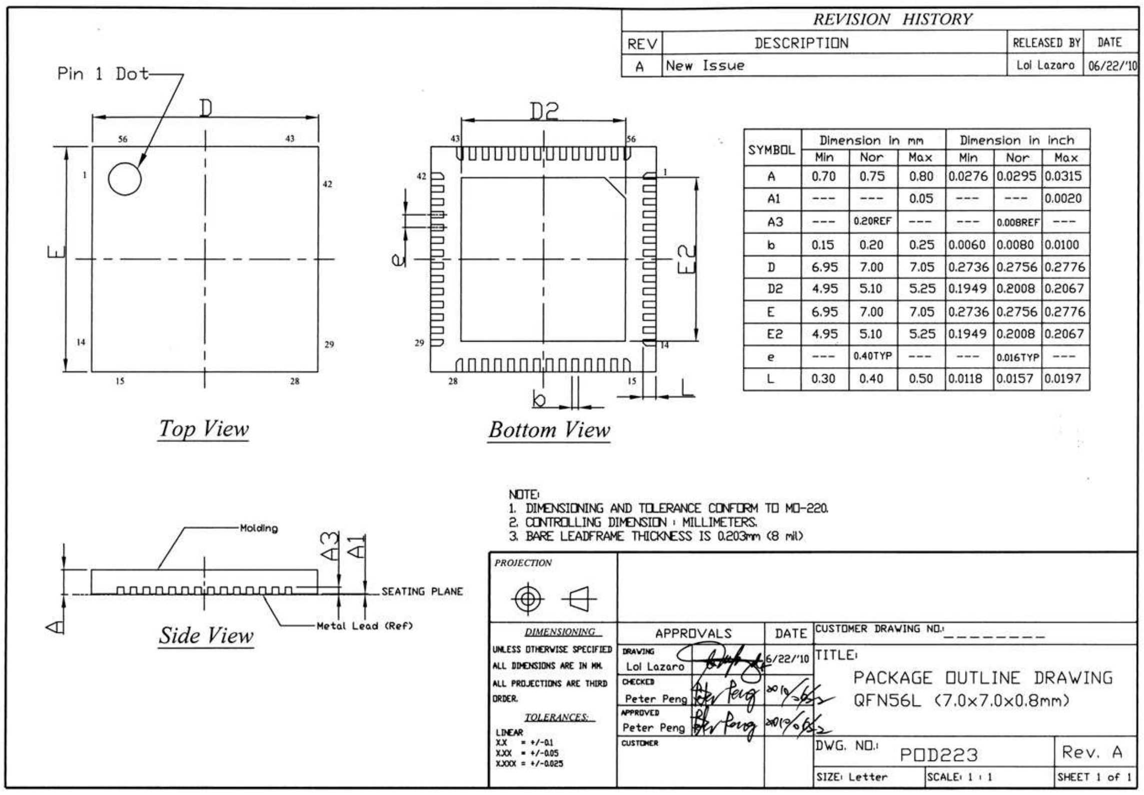
4.7.36 EEP_B_OPTION

Same function as EEP_X0_OPTION.

4.7.37 EEP_M0_B_SENSITIVITY

Used for to make adjustment the raw value. raw = raw ·(1 + sensitivity/128). Value range is -128 to 127.

5 Packaging information



REVISION HISTORY

REV	DESCRIPTION	RELEASED BY	DATE
A	New Issue	Loi Lazaro	06/22/'10

SYMBOL	Dimension in mm			Dimension in inch		
	Min	Nor	Max	Min	Nor	Max
A	0.70	0.75	0.80	0.0276	0.0295	0.0315
A1	---	---	0.05	---	---	0.0020
A3	---	0.20REF	---	---	0.008REF	---
b	0.15	0.20	0.25	0.0060	0.0080	0.0100
D	6.95	7.00	7.05	0.2736	0.2756	0.2776
D2	4.95	5.10	5.25	0.1949	0.2008	0.2067
E	6.95	7.00	7.05	0.2736	0.2756	0.2776
E2	4.95	5.10	5.25	0.1949	0.2008	0.2067
e	---	0.40TYP	---	---	0.016TYP	---
L	0.30	0.40	0.50	0.0118	0.0157	0.0197

NOTE:
1. DIMENSIONING AND TOLERANCE CONFORM TO MO-220.
2. CONTROLLING DIMENSION : MILLIMETERS.
3. BARE LEADFRAME THICKNESS IS 0.203mm (8 mil)

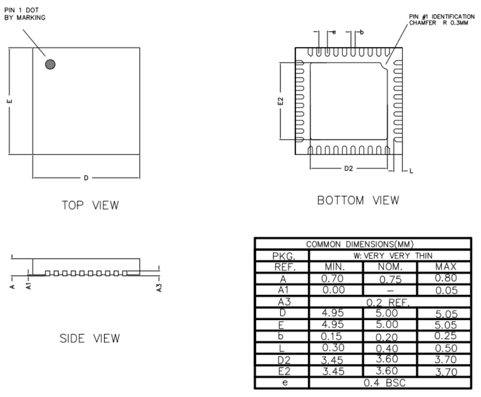
DIMENSIONING
UNLESS OTHERWISE SPECIFIED ALL DIMENSIONS ARE IN MM. ALL PROJECTIONS ARE THIRD ORDER.

TOLERANCES:
LINEAR
XX = +/-0.1
XXX = +/-0.05
XXXX = +/-0.025

APPROVALS		DATE
DRAWING Loi Lazaro	[illegible]	6/22/'10
CHECKED Peter Peng	[illegible]	2010/6/22
APPROVED Peter Peng	[illegible]	2010/6/22
CUSTOMER		

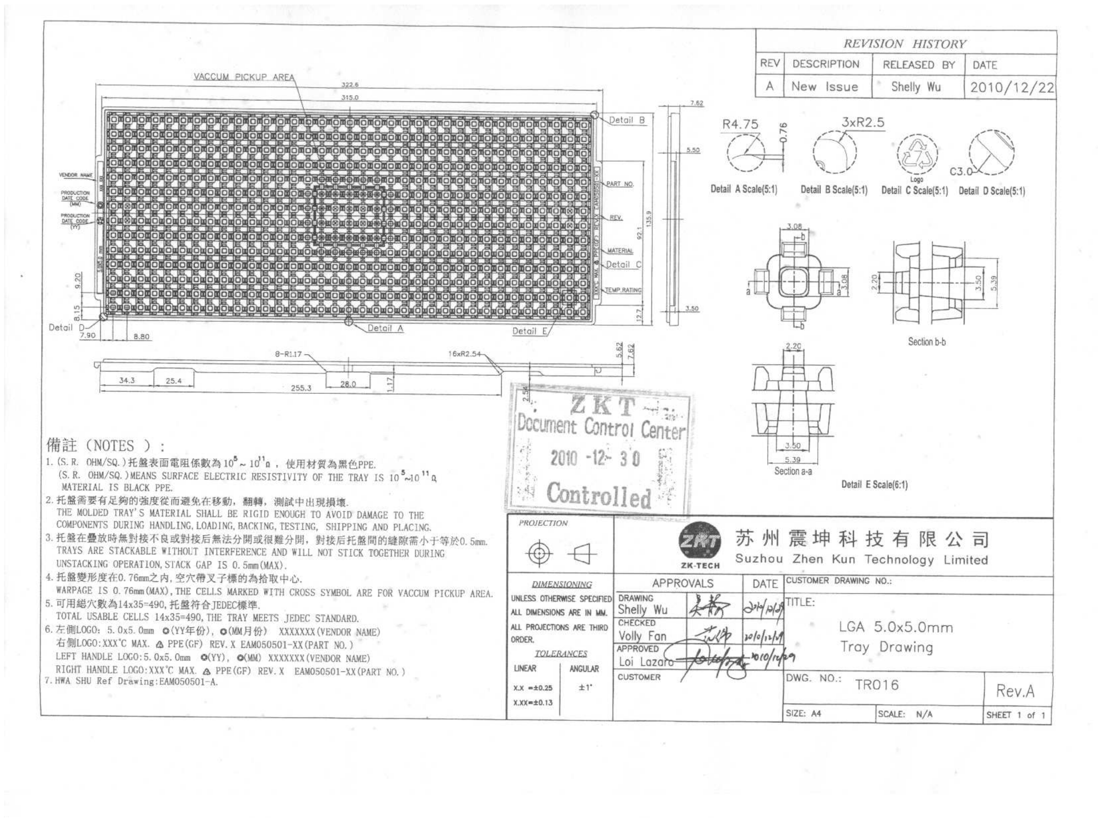
CUSTOMER DRAWING NO.: ________
TITLE: PACKAGE OUTLINE DRAWING QFN56L (7.0x7.0x0.8mm)
DWG. NO.: POD223 Rev. A
SIZE: Letter SCALE: 1 : 1 SHEET 1 of 1

Figure 77: Tango C44/C48 Packaging information



COMMON DIMENSIONS(MM)			
PKG.	W: VERY VERY THIN		
REF.	MIN.	NOM.	MAX
A	0.70	0.75	0.80
A1	0.00	–	0.05
A3	0.2 REF.		
D	4.95	5.00	5.05
E	4.95	5.00	5.05
b	0.15	0.20	0.25
L	0.30	0.40	0.50
D2	3.45	3.60	3.70
E2	3.45	3.60	3.70
e	0.4 BSC		

Figure 78: TANGO C32 Packaging information



REVISION HISTORY

REV	DESCRIPTION	RELEASED BY	DATE
A	New Issue	Shelly Wu	2010/12/22

備註（NOTES ）:

1. (S.R. OHM/SQ.)托盤表面電阻係數為$10^5 \sim 10^{11}\Omega$，使用材質為黑色PPE.
   (S.R. OHM/SQ.)MEANS SURFACE ELECTRIC RESISTIVITY OF THE TRAY IS $10^5 \sim 10^{11}\Omega$
   MATERIAL IS BLACK PPE.
2. 托盤需要有足夠的強度從而避免在移動，翻轉，測試中出現損壞.
   THE MOLDED TRAY'S MATERIAL SHALL BE RIGID ENOUGH TO AVOID DAMAGE TO THE COMPONENTS DURING HANDLING, LOADING, BACKING, TESTING, SHIPPING AND PLACING.
3. 托盤在疊放時無對接不良或對接后無法分開或很難分開，對接后托盤間的縫隙需小于等於0.5mm.
   TRAYS ARE STACKABLE WITHOUT INTERFERENCE AND WILL NOT STICK TOGETHER DURING UNSTACKING OPERATION, STACK GAP IS 0.5mm(MAX).
4. 托盤變形度在0.76mm之內, 空穴帶叉子標的為拾取中心.
   WARPAGE IS 0.76mm(MAX), THE CELLS MARKED WITH CROSS SYMBOL ARE FOR VACCUM PICKUP AREA.
5. 可用総穴數為14x35=490, 托盤符合JEDEC標準.
   TOTAL USABLE CELLS 14x35=490, THE TRAY MEETS JEDEC STANDARD.
6. 左側LOGO: 5.0x5.0mm ⭘(YY年份), ⭘(MM月份) XXXXXXX(VENDOR NAME)
   右側LOGO:XXX°C MAX. △ PPE(GF) REV.X EAM050501-XX(PART NO.)
   LEFT HANDLE LOGO:5.0x5.0mm ⭘(YY), ⭘(MM) XXXXXXX(VENDOR NAME)
   RIGHT HANDLE LOGO:XXX°C MAX. △ PPE(GF) REV.X EAM050501-XX(PART NO.)
7. HWA SHU Ref Drawing:EAM050501-A.

ZKT
Document Control Center
2010 -12- 30
Controlled

苏州震坤科技有限公司
Suzhou Zhen Kun Technology Limited

DIMENSIONING
UNLESS OTHERWISE SPECIFIED ALL DIMENSIONS ARE IN MM. ALL PROJECTIONS ARE THIRD ORDER.

TOLERANCES

LINEAR	ANGULAR
X.X =±0.25	±1°
X.XX=±0.13	

APPROVALS		DATE
DRAWING Shelly Wu	[illegible]	2010/12/29
CHECKED Volly Fan	[illegible]	2010/12/29
APPROVED Loi Lazaro	[illegible]	2010/12/29
CUSTOMER		

CUSTOMER DRAWING NO.:

TITLE: LGA 5.0x5.0mm Tray Drawing

DWG. NO.: TR016 Rev.A

SIZE: A4 SCALE: N/A SHEET 1 of 1

Figure 79: Tray C32

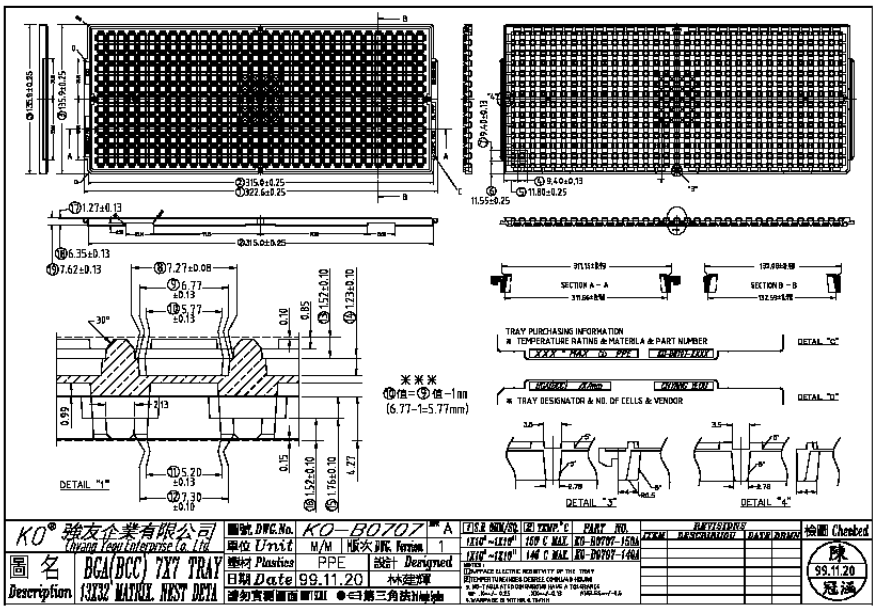


Figure 80: Tray C48/C44

Device Name	Device Type	Assembly Type	Pith	Tray
Tango C32	PIX97032SQ	QFN5*5 40L	0.5	490/Tray
Tango C44	PIX97044SQ	QFN7*7 56L	0.4	416/Tray
Tango C48	PIX97048SQ	QFN7*7 56L	0.4	416/Tray

Table 28: tray and tape & reel shipping products

6 Disclaimer