



Data sheet (Application note) #710

Tango C32/C44/C48

April 8, 2013

Abstract

This document describes Tango C32/C44/C48 usage.



Contents

1	Electrical specification of Tango C	7
1.1	Tango C series	7
1.2	Pin description	7
1.2.1	VCC	11
1.2.2	GND	11
1.2.3	RST	11
1.2.4	SDA	12
1.2.5	SCL	12
1.2.6	ATTB	13
1.2.7	CT	13
1.2.8	CSb	14
1.2.9	DI	14
1.2.10	CK	15
1.2.11	DO	15
1.2.12	Internal Tango sensor line	16
1.3	Absolute Maximum Ratings	17
1.4	Recommended Operation Conditions	17
1.5	Electrical interface	18
1.5.1	Interfacing with low supply host	18
1.5.2	States of ATTB line	19
1.5.3	Transition of ATTB line	19
1.6	Schematic for C32/C44/C48/C44+F32 circuit	23
1.7	Reset sources and sequences	27
1.7.1	Power on reset	27
1.7.2	Reset by RST pin	29
1.8	I ² C Protocol Specifications	31
1.8.1	I ² C features	31
1.8.2	I ² C address	31
2	Data Protocol	33
2.1	Introduction	33
2.1.1	Read operation	33
2.1.2	Write operation	34
2.2	MSI Registers	35
2.2.1	Endianness	35
2.2.2	Registers organization	35
2.2.3	RAW_CTRL write&read	40
2.2.4	Power_mode register	43
2.2.5	INT_mode register	43
2.3	Details about coordinates report and finger ID	44
2.3.1	Double buffering and atomic access	44
2.3.2	Status registers	45
2.3.3	Difference in finger ID and finger slot	45
2.4	Power management	49
2.4.1	Active mode	49
2.4.2	Idle mode	49
2.4.3	Sleep mode	49
2.4.4	Power consumption	50
2.4.5	Power mode diagram	51

2.5	Special operations	52
2.5.1	Normal Mode	52
2.5.2	EEPROM read operation	52
2.5.3	Calibration	52
2.5.4	Border tracking calibration	53
2.5.5	CRC checksum	55
2.5.6	Soft reset (jump to bootloader)	55
2.6	Auto calibration	55
2.6.1	Introduction	55
2.6.2	Offset stored in flash	55
2.6.3	When the offset must be stored in the flash ?	56
2.6.4	What is the purpose of the auto calibration?	57
2.6.5	Summary of the auto calibration	58
2.7	Autocalibration for buttons	58
2.7.1	Schematic change for the external Lift for button autocalibration	58
2.7.2	Scan procedure for Autocalibration Button	59
2.7.3	Configuration of the button autocalibration EEPROM parameters	59
2.8	Coordinates characteristics	59
3	Bootloader	61
3.1	Introduction	61
3.1.1	Bootloader general flow diagram	61
3.1.2	Bootloader Application FW download flow diagram	63
3.1.3	Bootloader EEPROM download flow diagram	64
3.2	Switching to bootloader	65
3.3	Bootloader read operation	65
3.4	Bootloader write operation	66
3.5	File format	68
3.5.1	Frame format	68
3.5.2	File example	68
3.6	Error handling	69
3.7	Example	70
4	EEPROM parameters	71
4.1	EEPROM features	71
4.2	Structure members	71
5	Packaging information	72
6	Disclaimer	76

Revision history

Version	Content	Pages	Date	Author	Reviewed
0	Initial release	59	23rd of May 2011	Jeff Apollo	LPO
1	<p>Add figure 2: C48 pins configuration.</p> <p>Add table 2: C48 pins description.</p> <p>Add figure 1: C32 pins configuration.</p> <p>Add table ??: C32 pins description.</p> <p>Add figure 26: C32 circuit</p> <p>Add figure 26: C44 circuit</p> <p>Add figure 27: C48 circuit</p> <p>Add figure 28: C44+C32 circuit</p> <p>And other information about C48</p> <p>Add the figure 31 figure 33 show in idle and active mode</p> <p>Add the figure 72 for the C32 package</p> <p>Add figure ??: for EEP_M0_X_THR_ONEFINGER parameter</p> <p>Redraw the figure ??: in EEP_M0_X_THR_ONEFINGER section</p>	67	26 of May 2011	Jeff Apollo	LPO
2	Modify the figure 7, figure 8 and figure 16	67	10 of june 2011	Jeff	LPO Marc
3	<p>Modify the figure 25 figure 26 , figure 27 and figure 28</p> <p>update EEPROM content</p> <p>figure ??</p> <p>figure ??</p> <p>figure ??</p>	70	22th June 2011	Jeff apollo	
4	<p>Add the table 6</p> <p>Add the table 7</p>	70	20th July 2011	Jeff	Jeffery Alex Janson
5	<p>Modify the figure 25 figure 26 figure 27 figure 28</p> <p>Add the Principle of the buttons</p>	73	29th July 2011	Jeff	nicole
6	<p>Modify the Tango Core button circuits</p> <p>table 22</p> <p>Add table 27</p> <p>figure 74</p> <p>figure 75</p> <p>EEPROM parameter</p>	73	08/25/11	Apollo	Marc Angela



PIXCIR PROPERTY FOR ZHONGFU REFERENCE ONLY



Version	Content	Pages	Date	Author	Reviewed
7	For 2 fingers FW only Modify the figure 26 figure 26 figure 27 figure 67 figure 26 figure 27 figure 28 figure 67 add Idle_freq and Auto_idle_delay in table 15 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
8	figure 51 figure ?? figure ?? figure ?? update IIC table 14 update power consumption data modify power mode chapter add eep parameters table	90	2011-12-19	Apollo	Dominik Marc
8.1	figure 29 figure 66 Correct "Bootloader flow diagram" mistake Removed unuseful eep parameters: EEP_EXTRA_BORDER_CORR EEP_CORRX EEP_CORRY	96	2012-03-27	Apollo	Dominik Marc
8.2	Modify Tang C max scan pin numbers for Active area	96	2012-Sep-20	Apollo	Marc
8.3	Add new package C48S LGA 5*5mm	98	2012-Otc-23	Apollo	Cristina
8.4	modify MP9.0 dominik version	102	2012-Nov-05	Apollo	
9.0	update BL description to BL V48	102	2013-Feb-01	Dominik	Romain
9.2	EEPROM layout upgrade to FW MP9.2, upgrade to BL V49	104	2013-Feb-06	Dominik	
9.3	EEPROM layout upgrade to FW MP9.3, added schematic for Button Autocalibration	108	2013-Mar-11	Dominik	
9.4	EEPROM layout upgrade to FW MP9.4 (based on Tan- goC_EEPROM_layout_V4.3.xls), update the MSI table (based on i2c_v18 .xls) and several small corrections/updates	113	2013-Mar-13	Ruben	
10.0	Added and updated parameter information, moved EEPROM description to extra AppNote 716 / 76	76	2013-Apr-05	Dominik	

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 otherwise in this document, the description is valid for the C32, C44 and C48.

1.2 Pin description

Figure 1, figure 2, figure 3 and figure 4 show one chip solution (TANGO C32 40NQ)/(TANGO C44 56NQ)/(TANGO C48 56NQ)/(TANGO C48S LGA56) pins configuration. The table 1, table 2, table 3 and table 4 are summarize the pin functions of Tango C32/C44/C48/C48S.

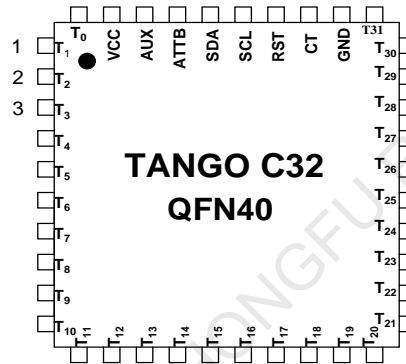


Figure 1: C32 pins configuration (top view)

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

Table 1: C32 pins description

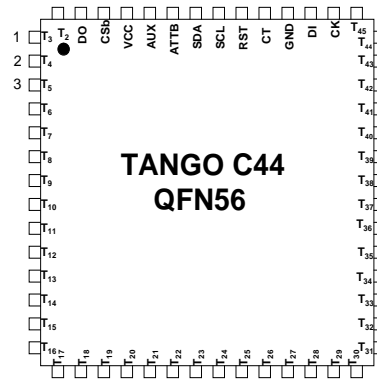


Figure 2: C44 pins configuration (top view)

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..T₄₅}	44 Tango Sensor lines

Table 2: C44 pins description

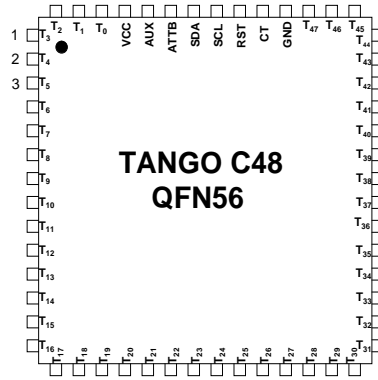


Figure 3: C48 pins configuration (top view)

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

Table 3: C48 pins description

Figure 4 shows one chip solution TangoC48S (LGA56 5X5mm). The table 4 summarizes the pin functions of Tango C48S.

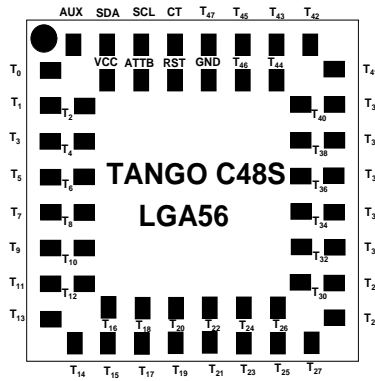


Figure 4: C48S pins configuration (top view)

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

Table 4: C48S pins description

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 5: Reset input, active high, with low threshold voltage. A high level above $V_{RST,HI}$ for a duration longer than $T_{RST,IN}$ 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 5 and illustrated in figure 6.

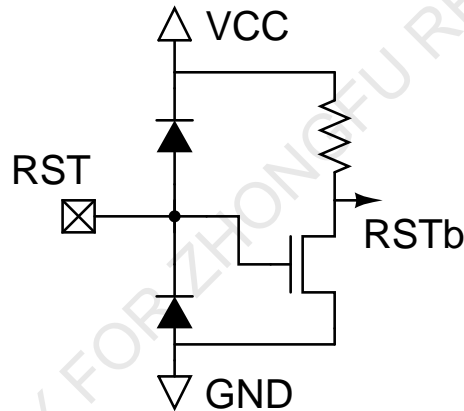


Figure 5: RST pin circuit

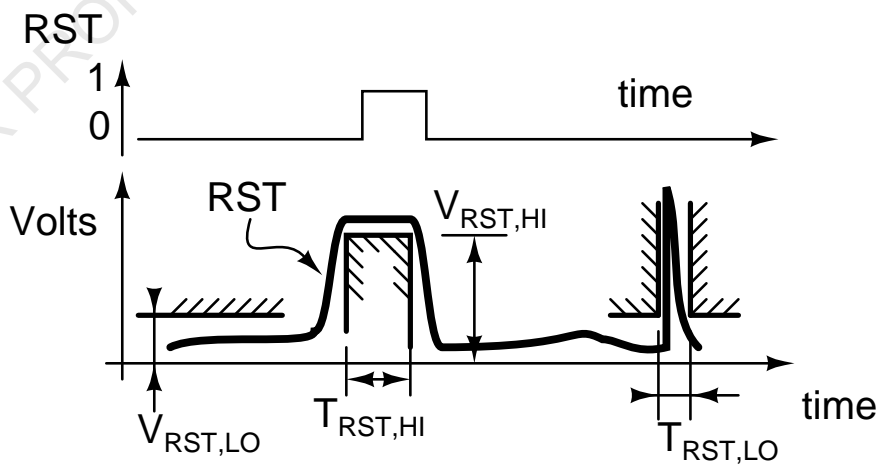


Figure 6: Reset timing

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

Table 5: RST pin characteristics

1.2.4 SDA

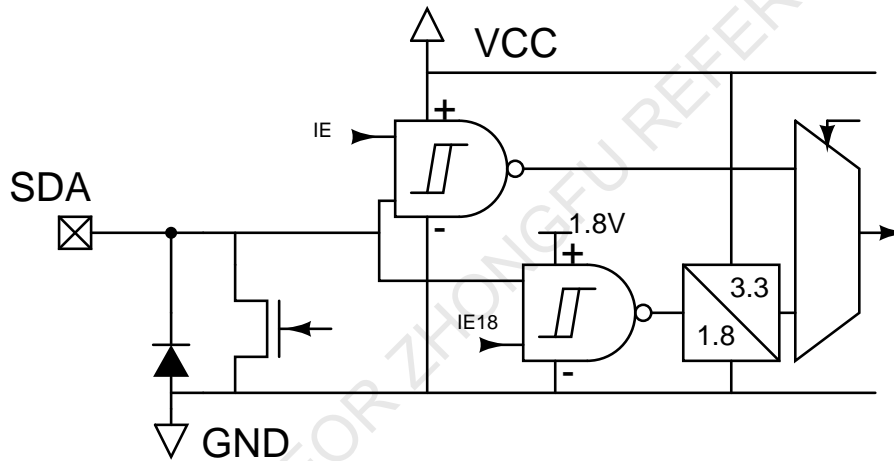


Figure 7: SDA pin circuit

Figure 7: 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

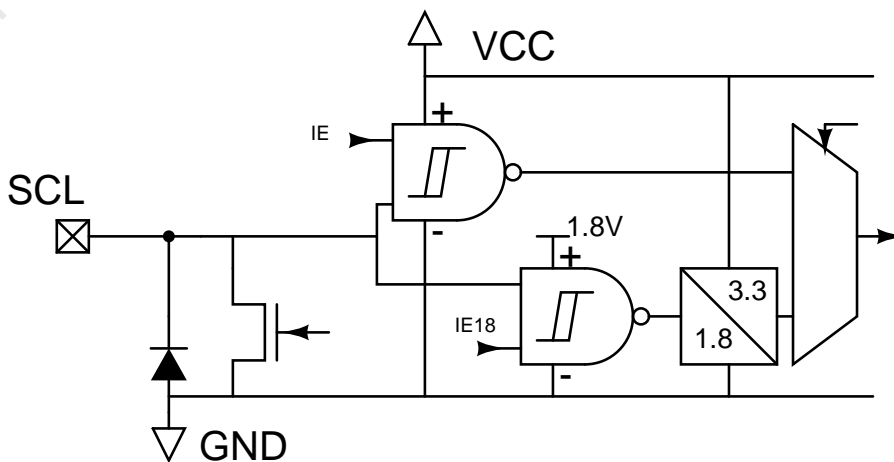


Figure 8: SCL pin circuit

Figure 8 : 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

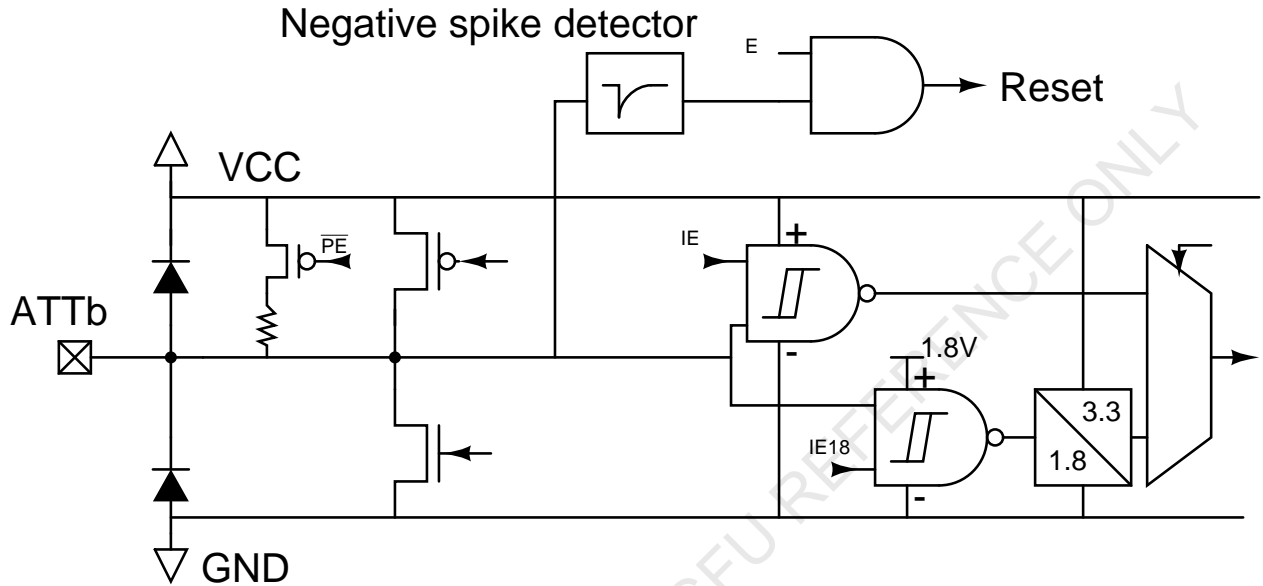


Figure 9: ATTB pin circuit

Figure 9: 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

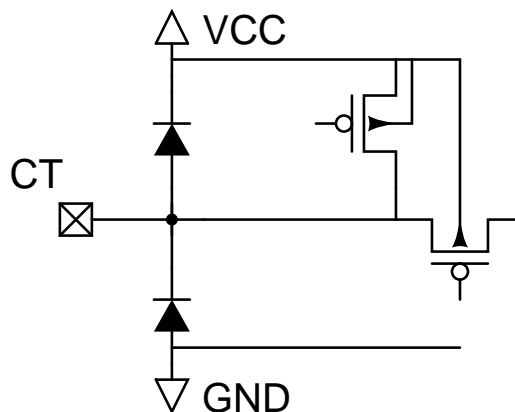


Figure 10: CT pin circuit

Figure 10: 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

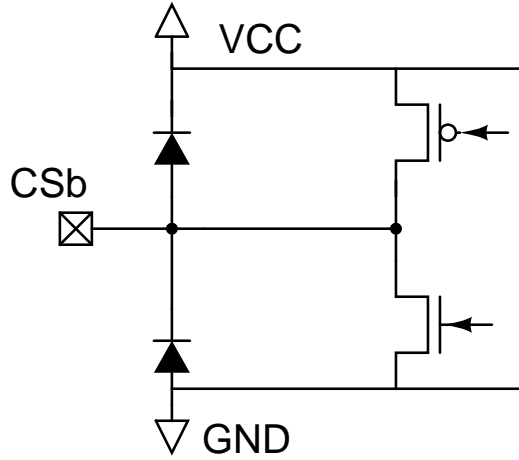


Figure 11: CSb pin circuit

CSb is only available with Tango C44. Figure 11: External Tango chip select. This output can drive a Tango F32 or F48 CSb input pad.

1.2.9 DI

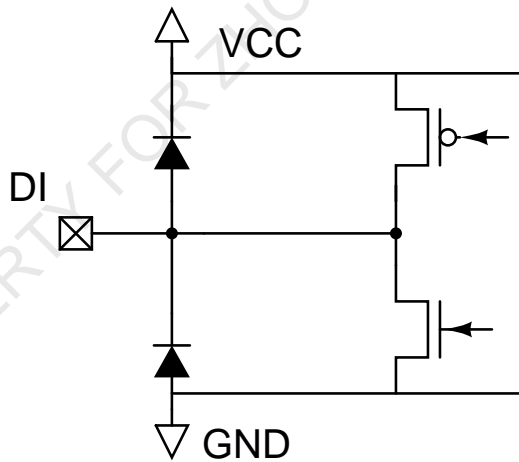


Figure 12: DI pin circuit

DI is only available with Tango C44. Figure 12: External Tango master to slave data. This output can drive a Tango F32 or F48 DI input pad.

1.2.10 CK

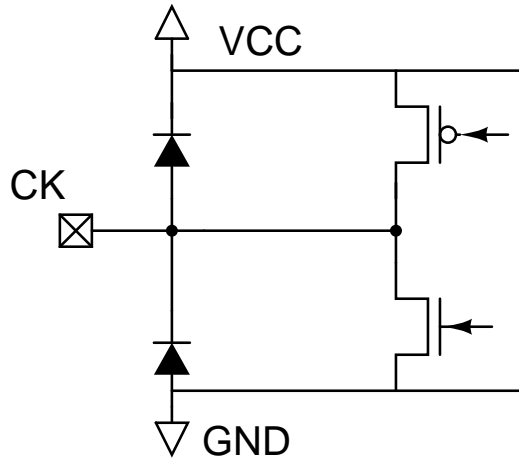


Figure 13: CK pin circuit

CK is only available with Tango C44. Figure 13: External Tango master clock. This output can drive a Tango F32 or F48 CK input pad.

1.2.11 DO

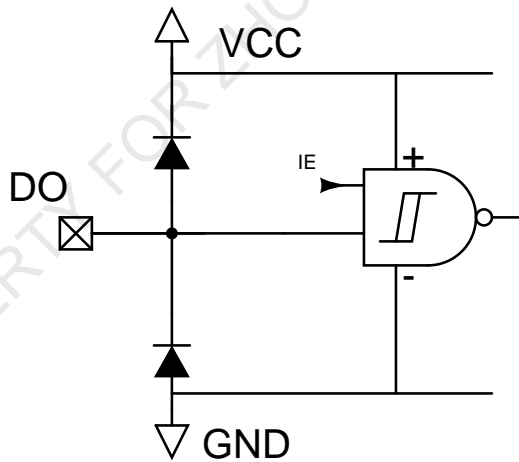


Figure 14: DO pin circuit

DO is only available with Tango C44. Figure 14: 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

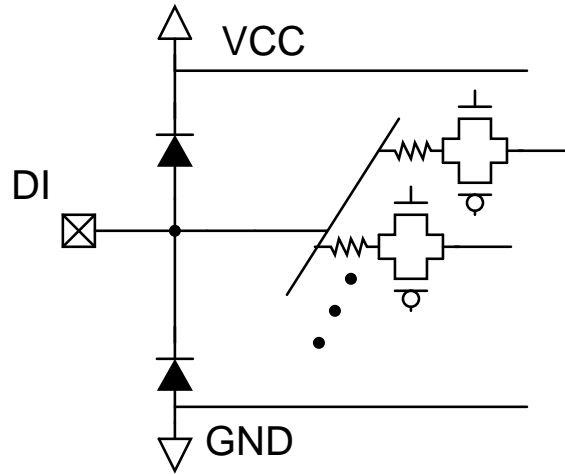


Figure 15: sense pin circuit

Figure 15: Tango C44 has 44 sensor lines($T_2 \dots T_{45}$). Tango C48 has 48 sensor lines($T_0 \dots T_{47}$), and Tango C32 has 32 sensor lines($T_0 \dots T_{31}$).

PIXCIR PROPERTY FOR ZHONGFU REFERENCE ONLY

1.3 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	°C
Tstg	Storage temperature	-55 to 125	°C
V _{ESD}	Electrostatic discharge (HBM model)	4'000	V

Table 6: Absolute Maximum Ratings

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

MSL: Level 3

1.4 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		-	-	24	MHz
-	CPU clock frequency		-	-	24	MHz
-	Operating temperature		-30	-	85	°C

Table 7: Recommended Operation Conditions

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 I²C signals SDA and SCL, as shown in figure 16. The MSI controller is allowed to set the ATTB line .

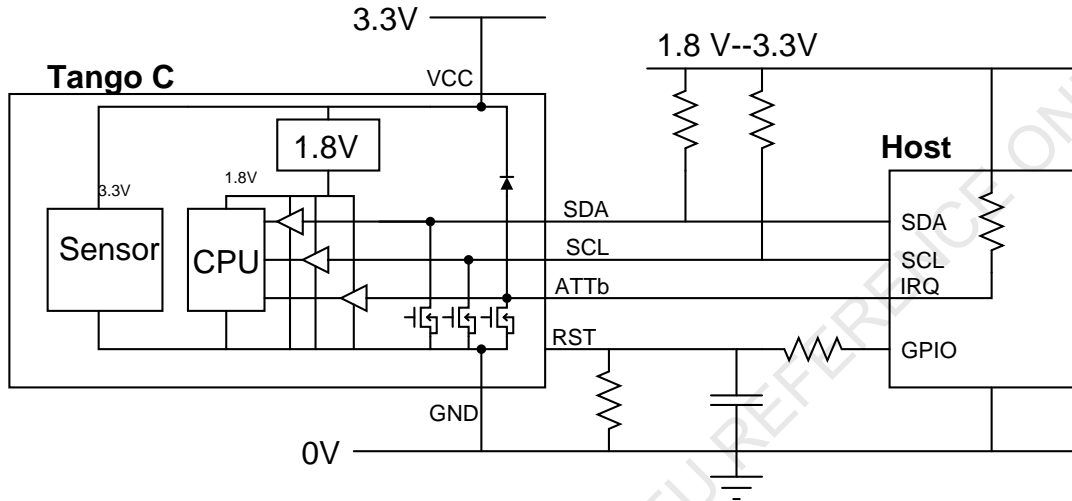


Figure 16: Example of interface with low voltage IIC bus

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 `EEP_IO_SWITCHING_LEVEL` can switch 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 I²C slave device and fully complies with I²C addressing and usual I²C hand shake protocol. As such, a MSI controller is suitable in an bus shared with other I²C 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 17 and figure 18.

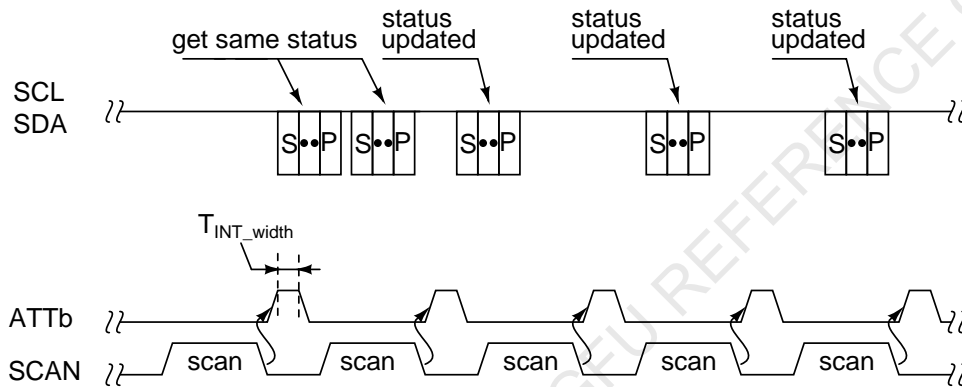


Figure 17: ATTB line pull up by slave (INT_POL=1,INT_MODE=00 in the INT mode register)

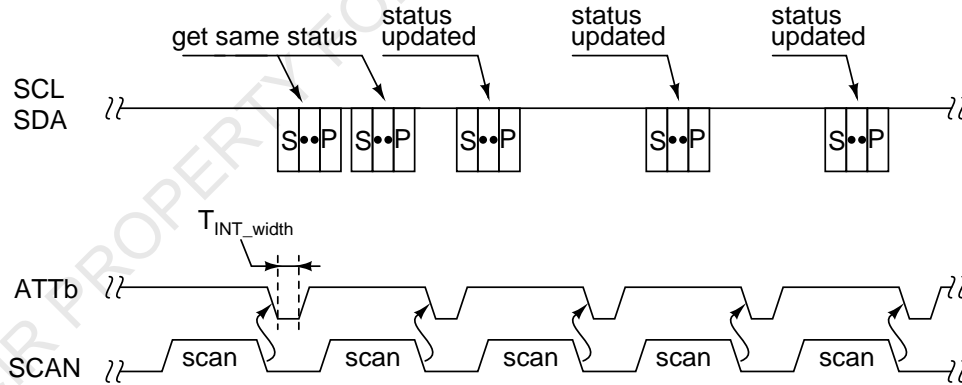


Figure 18: ATTB line pull down by slave (INT_POL=0, INT_MODE=00 in the INT mode register)

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 19. 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 20.

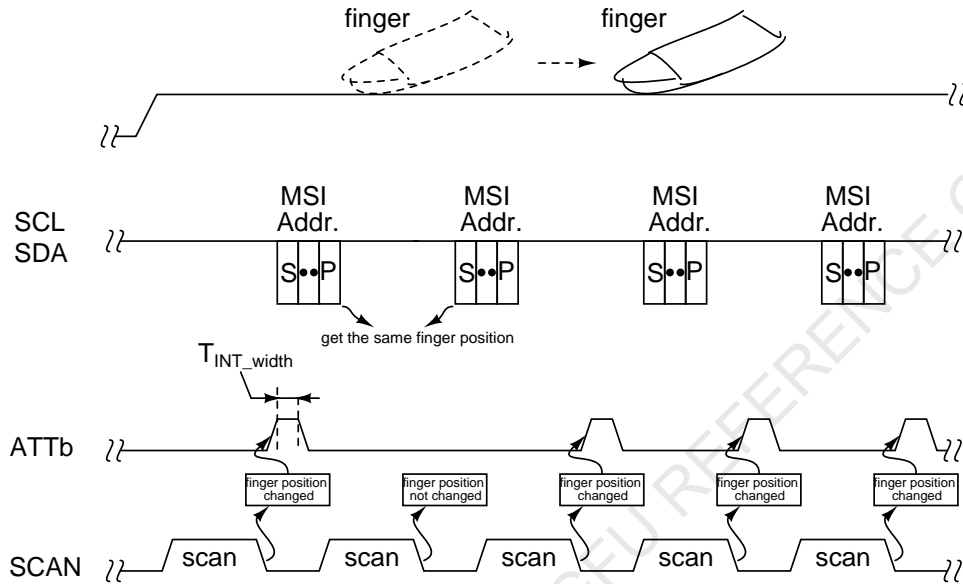


Figure 19: ATTB line pull up when finger moving (INT_POL=1, INT_MODE=01 in the INT mode register)

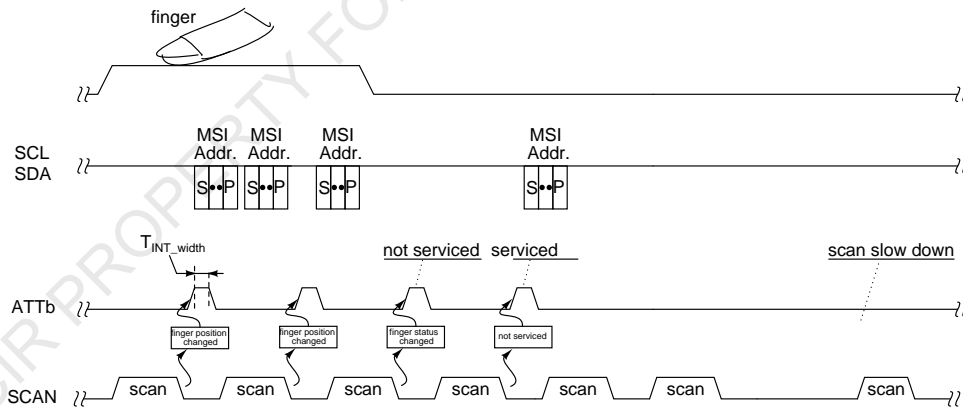


Figure 20: ATTB line will stop pulse when finger leaves and master has acknowledged the situation (INT_POL=1 in the INT mode register)

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 21. 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 22 .

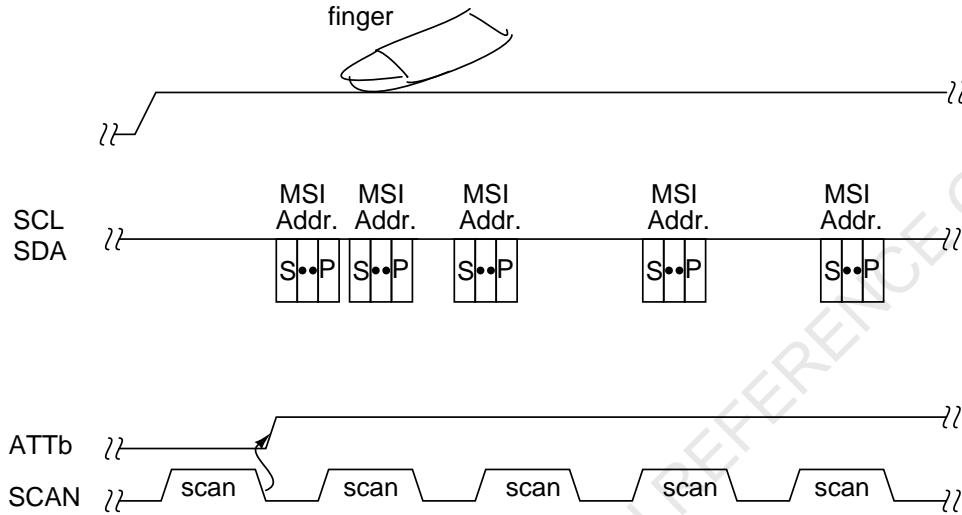


Figure 21: ATTB line pull up when finger touch (INT_POL=1, INT_MODE=10 in the INT mode register)

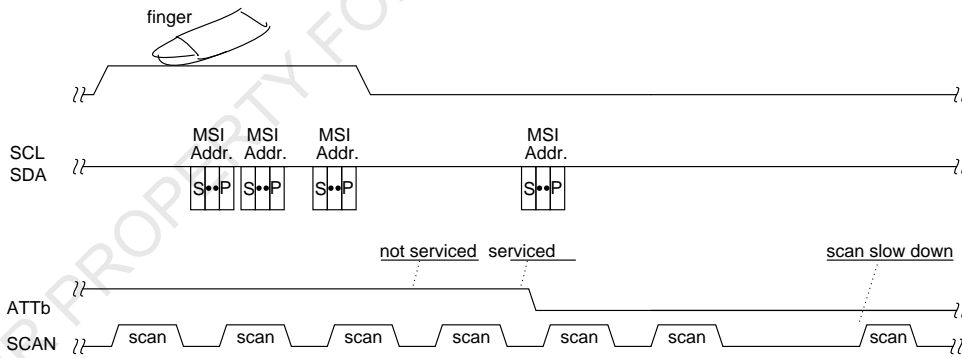


Figure 22: ATTB line will reset level when finger leaves and master has acknowledged the situation (INT_POL=1 in the INT mode register)

The only difference between INT_Mode=10 and INT_Mode=11 is that an ATTB pulse is sent instead of setting the level.

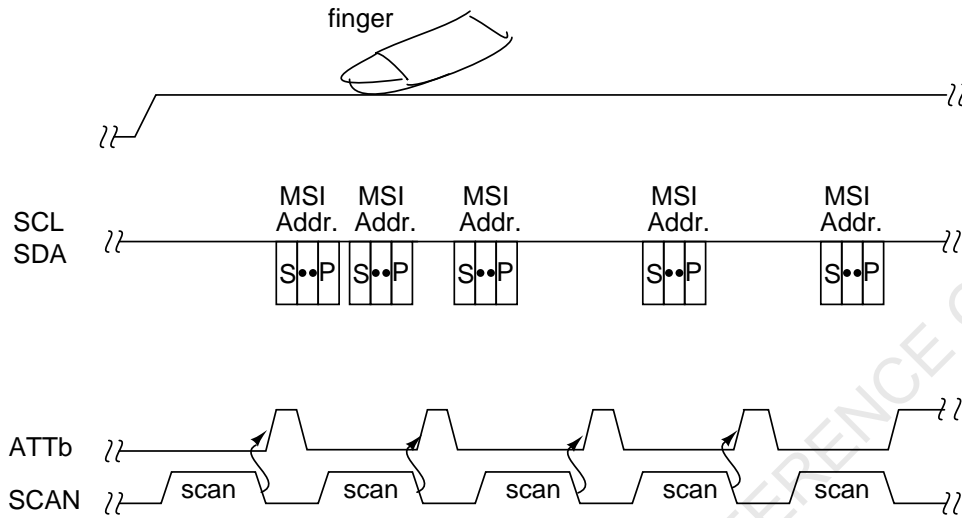


Figure 23: ATTB line pull up when finger touch (INT_POL=1, INT_MODE=11 in the INT mode register)

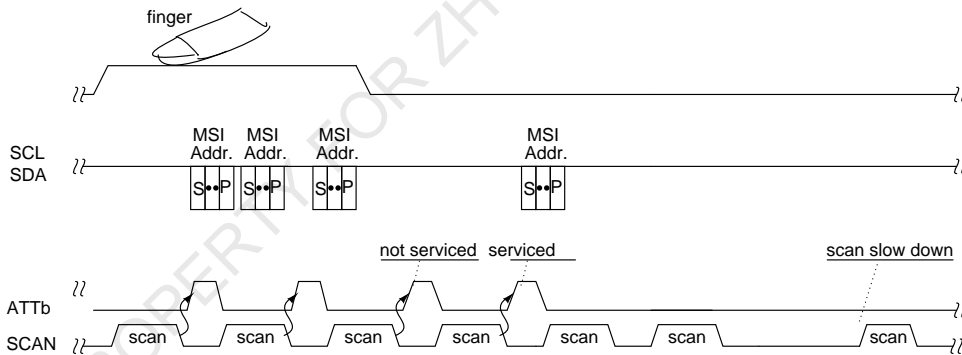


Figure 24: ATTB line will stop pulse when finger leaves and master has acknowledged the situation (INT_POL=1 in the INT mode register)

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

Pinmaps design ruler:

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

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

Buttons: Maximum 8 buttons, Minimum buttons is 2.

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

If there is enough sensor lines, we suggest two dummy pins for each axis and one dummy for buttons.

Y axis \ X axis	21	22	23	24	25	26	27	28	29	30	31	32	33
21	484	506	528	550	572	594	616	638	660	682	704	726	748
22	506	529	552	575	598	621	644	667	690	713	736	759	782
23	528	552	576	600	624	648	672	696	720	NOT OK			816
24	550	575	600	625	650	675	700	725	750				850
25	572	598	624	650	676	702	728	754	780				884
26	594	621	648	675	702	729	756	783	810				918
27	616	644	672	700	728	756	784	812	840	868	896	924	952
28	638	667	696	725	754	783	812	841	870	899	928	957	986
29	660	690	720	750	780	810	840	870	900	930	960	990	1020
30	682	713	744	775	806	837	868	899	930	961	992	1023	1054
31	704	736	768	800	832	864	896	928	960	992	1024	1056	1088
32	726	759	792	825	858	891	924	957	990	1023	1056	1089	1122
33	748	782	816	850	884	918	952	986	1020	1054	1088	1122	1156
34	770	805	840	875	910	945	980	1015	1050	1085	1120	1155	1190
35	792	828	864	900	OK			1044	1080	1116	1152	1188	1224
36	814	851	888	925				1073	1110	1147	1184	1221	1258
37	836	874	912	950				1102	1140	1178	1216	1254	1292
38	858	897	936	975				1131	1170	1209	1248	1287	1326
39	880	920	960	1000	1040	1080	1120	1160	1200	1240	1280	1320	1360
40	902	943	984	1025	1066	1107	1148	1189	1230	1271	1312	1353	1394
41	924	966	1008	1050	1092	1134	1176	1218	1260	1302	1344	1386	1428
42	946	989	1032	1075	1118	1161	1204	1247	1290	1333	1376	1419	1462
43	968	1012	1056	1100	1144	1188	1232	1276	1320	1364	1408	1452	1496
44	990	1035	1080	1125	1170	1215	1260	1305	1350	1395	1440	1485	1530

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) double check above figure		
Twin Tango	C44+F32	76	74(X+Y) double check above figure		
Twin Tango	C44+F48	76	74(X+Y) double check above figure		

Table 8: pinmap design rule

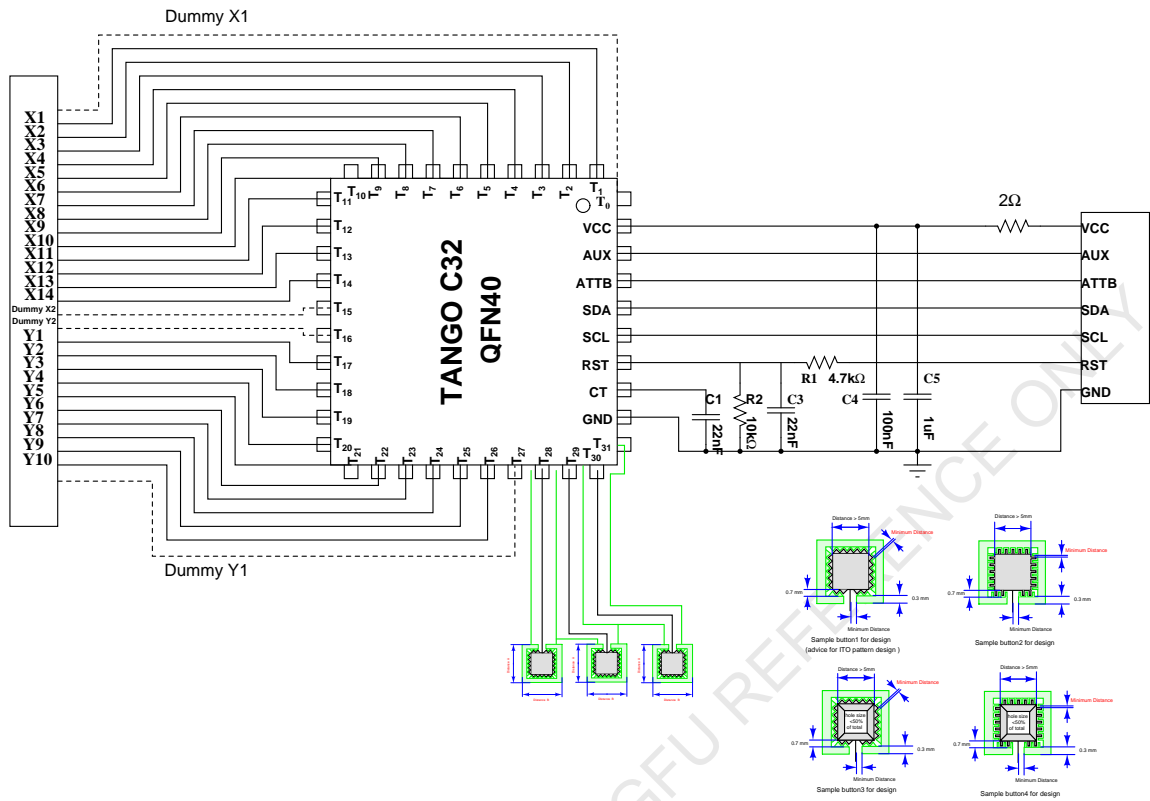


Figure 25: tango_C32_circuit

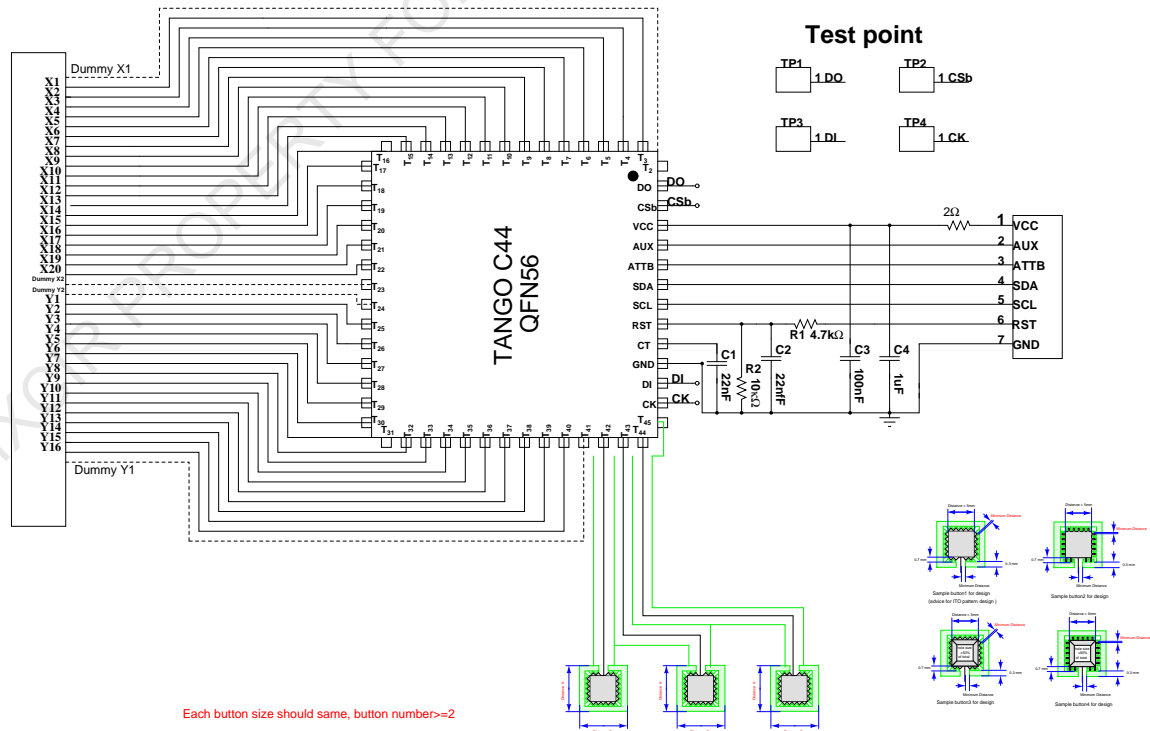


Figure 26: tango_C44_circuit

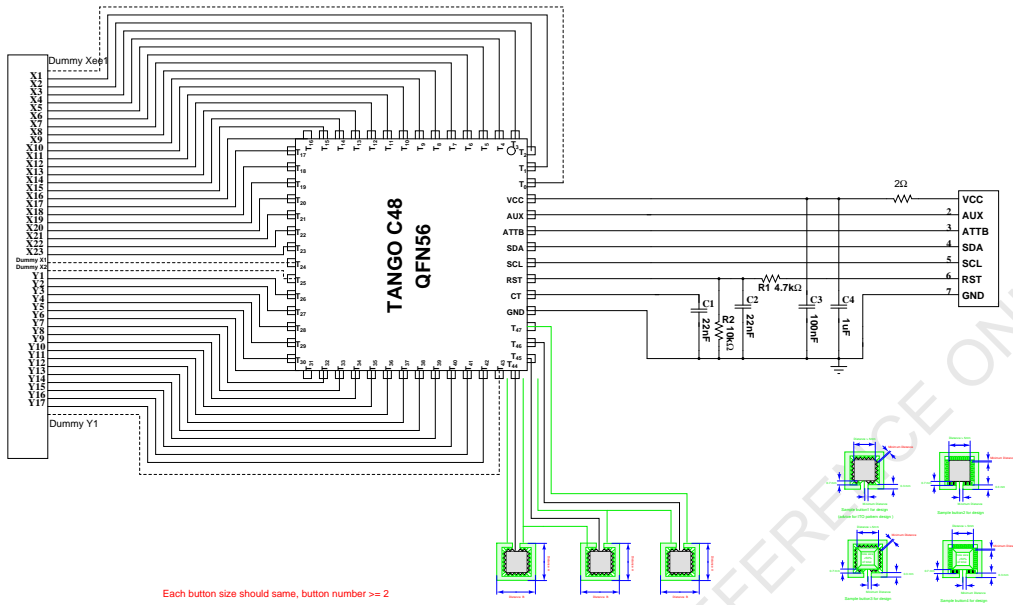


Figure 27: tango_C48_circuit

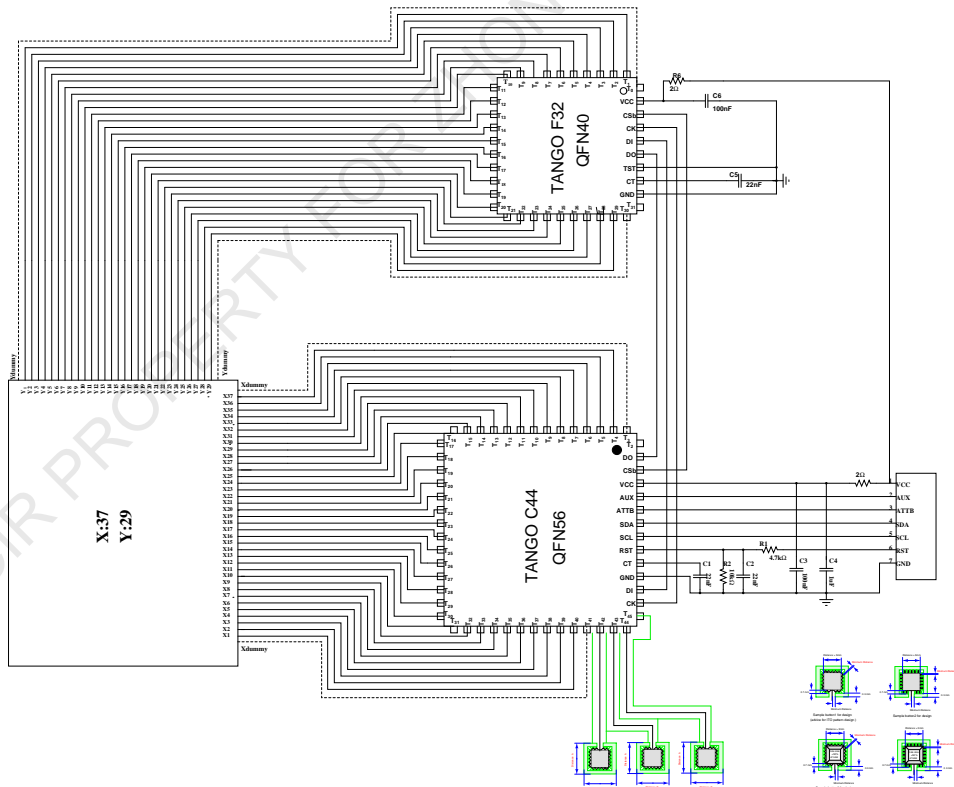


Figure 28: tango_C44+F32_circuit

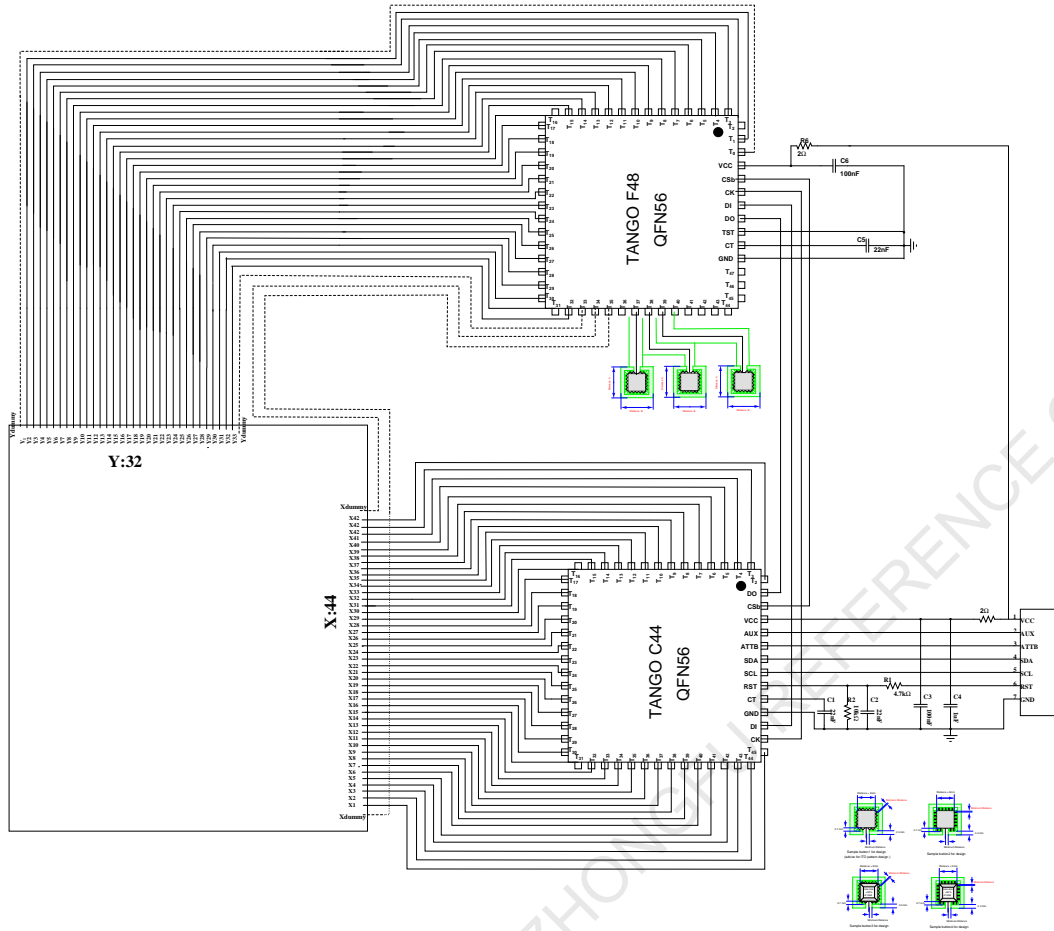


Figure 29: tango_C44+F48_circuit

1.7 Reset sources and sequences

The specification of the reset and power modes are summarized in tables 9, 10, 11.

1.7.1 Power on reset

Reset pin tied to ground with 10kΩ resistance, result shown in figure 30.

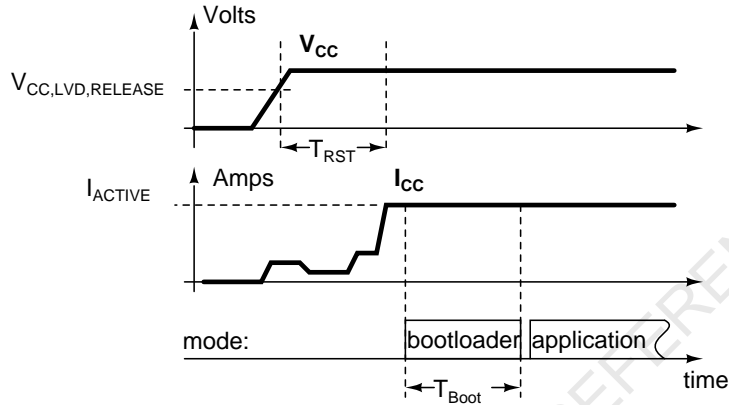


Figure 30: Power on sequence-1

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		tdb		V
$T_{POR,OFF}$	Power on reset turn off time	0.4		0.7	ms

Table 9: Power on reset

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 $V_{CC,LVD,SET}$ ensuring a reset		tdb		s

Table 10: Low level detector

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, 24MHz CPU		6	9	mA
I_{WAIT}	Wait mode, no touch sensor connected, VCC =3.3V, 24MHz CPU		3		mA
I_{IDLE}	Sleep mode, no touch sensor connected, VCC =3.3V, 24MHz CPU. Consumption includes periodical wake up: 60ms sleep and 50us active execution (1000 cycles).		45		uA
I_{SLEEP}	Sleep mode, no touch sensor connected, VCC =3.3V, 24MHz CPU		1		uA
I_{RESET}	Reset mode, no touch sensor connected, VCC =3.3V, 24MHz CPU		2		mA

Table 11: Power consumption, for the CPU

1.7.2 Reset by RST pin

Two cases are distinguished:

1. Reset while Tango C is in active mode, shown in figure 32.
2. Reset while the Tango C is in sleep mode, shown in figure 34.

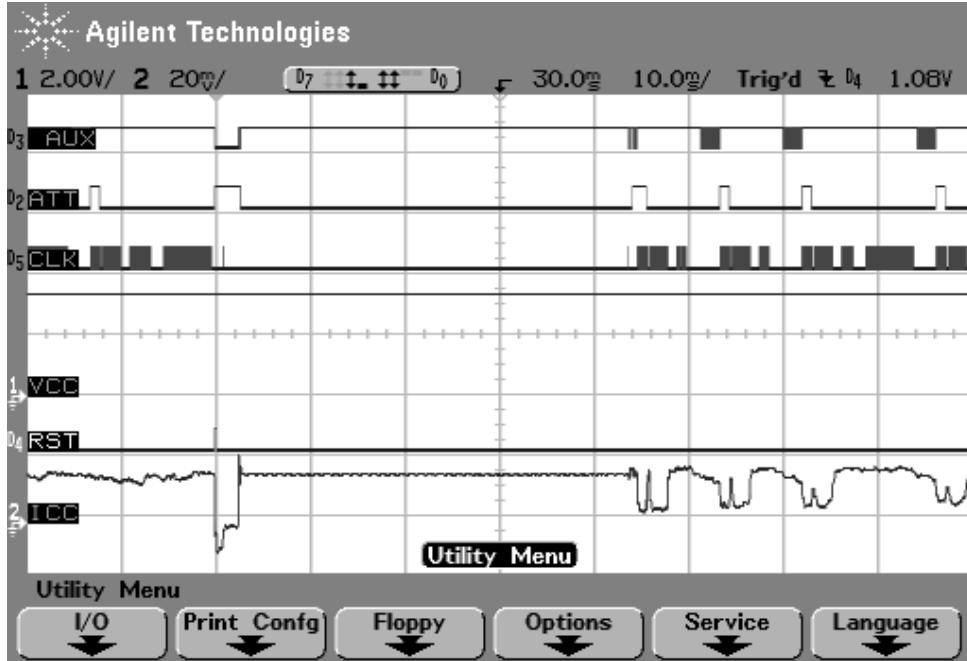


Figure 31: The oscilloscope shows the reset when in active

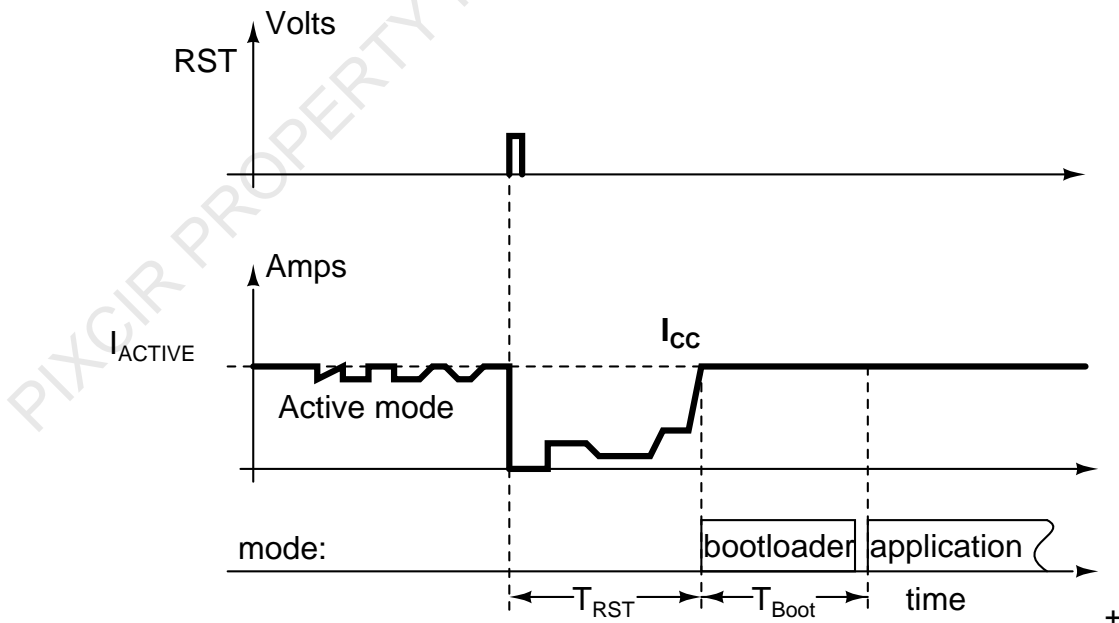


Figure 32: Power on sequence-2 (reset controlled by Host MCU when in active)

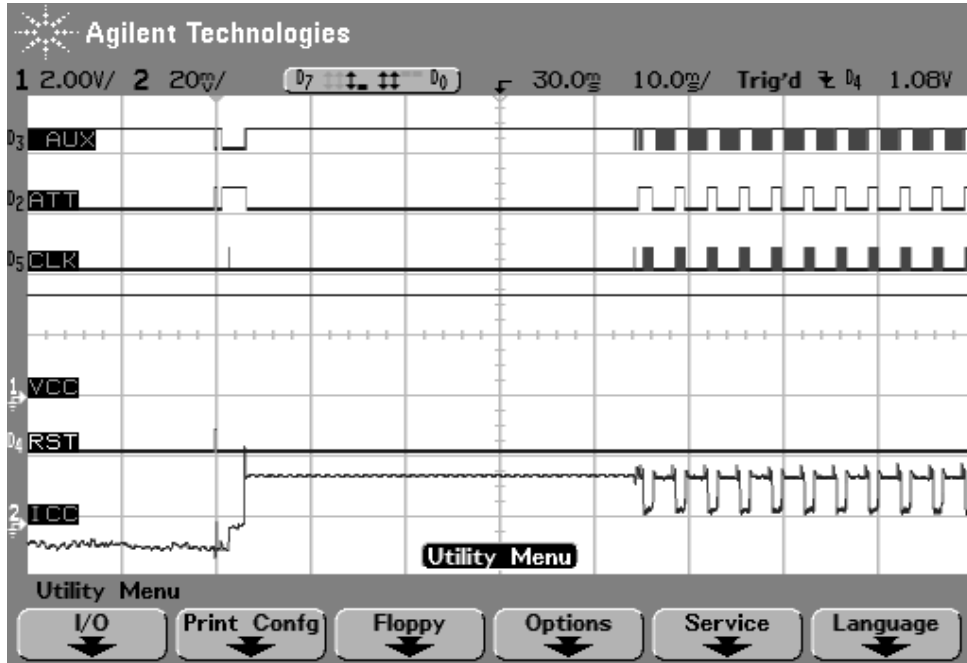


Figure 33: The oscilloscope shows the rest when in sleep

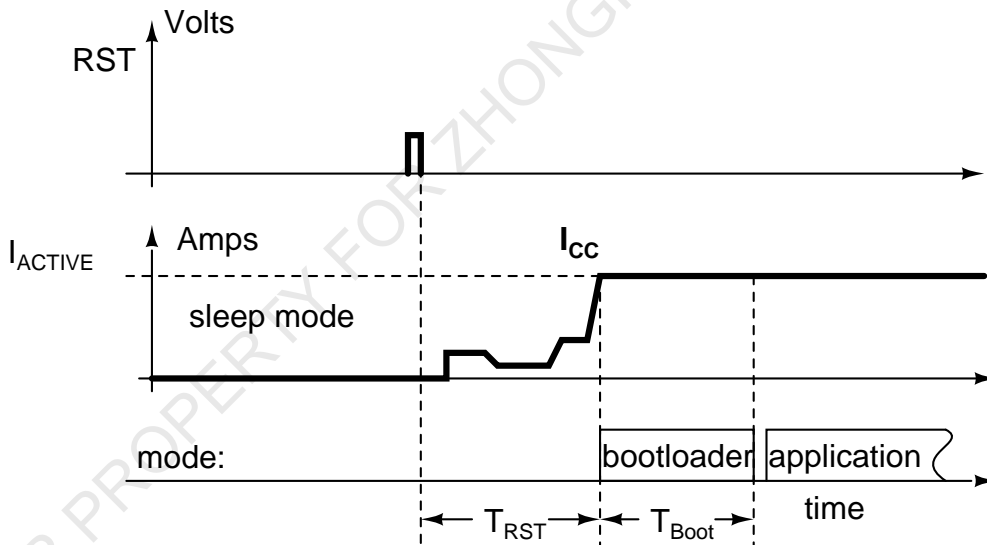


Figure 34: Power on sequence-3 (reset controlled by Host MCU when in sleep)

1.8 I²C Protocol Specifications

1.8.1 I²C 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 I²C master can't finish 1byte data in 100ms, I²C slave will restart. Tango C44 operates only as a slave device. The I²C 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 implement different I²C touch protocols. The timings for example are shown in table 12 and figure 35.
5. I²C 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 36 for an example.

1.8.2 I²C address

1. MSI device address = 0x5C.

Symbol	Parameter	Min	Typ	Max	Unit
T_{LOW}	IIC clock low time	$2 \cdot T_{CPU}$			
T_{HIGH}	IIC clock high time	$2 \cdot T_{CPU}$			
$T_{HD,STA}$	IIC clock hold time	$2 \cdot T_{CPU}$			
$T_{SU,STA}$	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	$9 \cdot T_{CPU}$			
$T_{VD,DAT}$	I C data valid after clock change, when data is driven by slave side	$9 \cdot T_{CPU}$			
T_{TCPU}	CPU master clock period			55	ns

Table 12: I²C timing

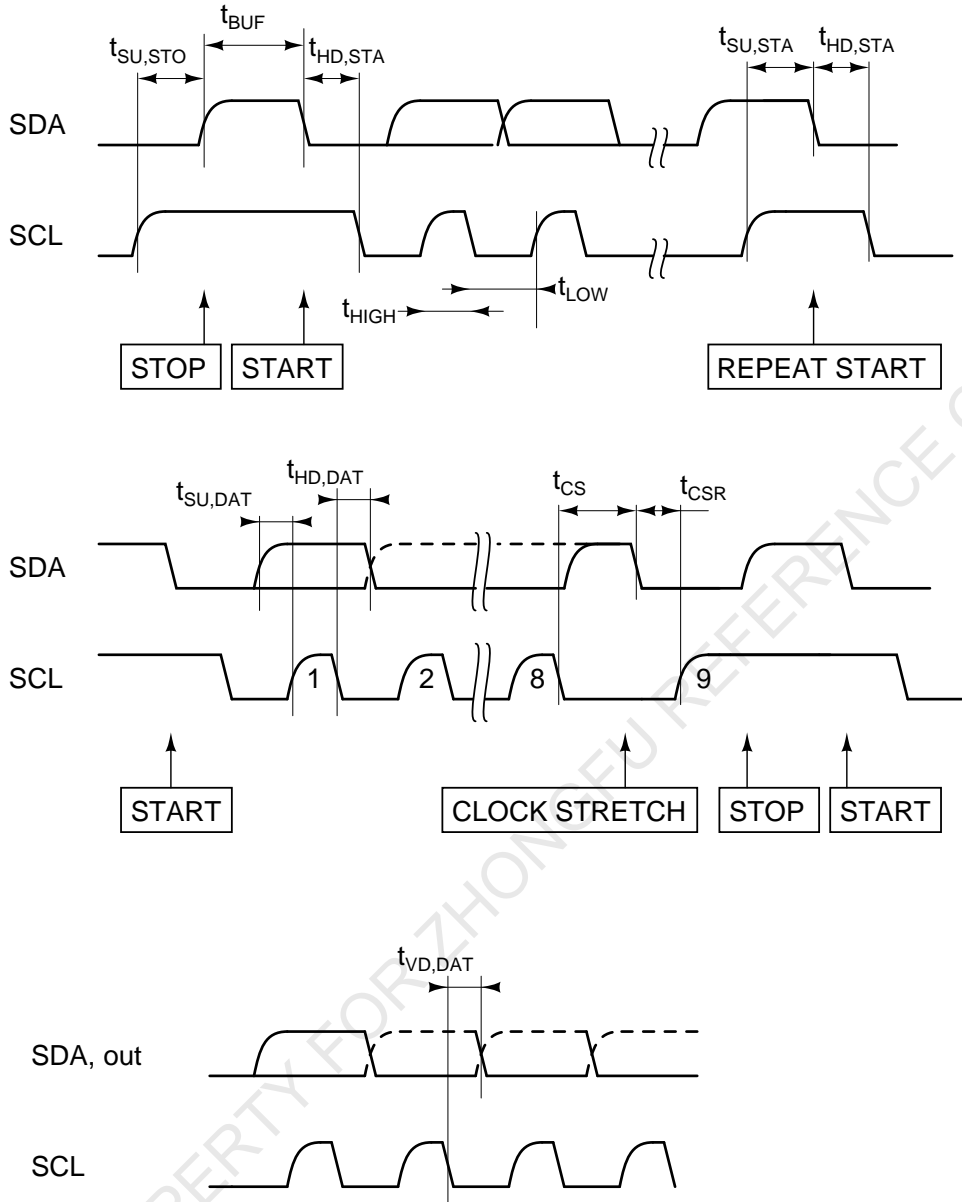


Figure 35: I²C timing example

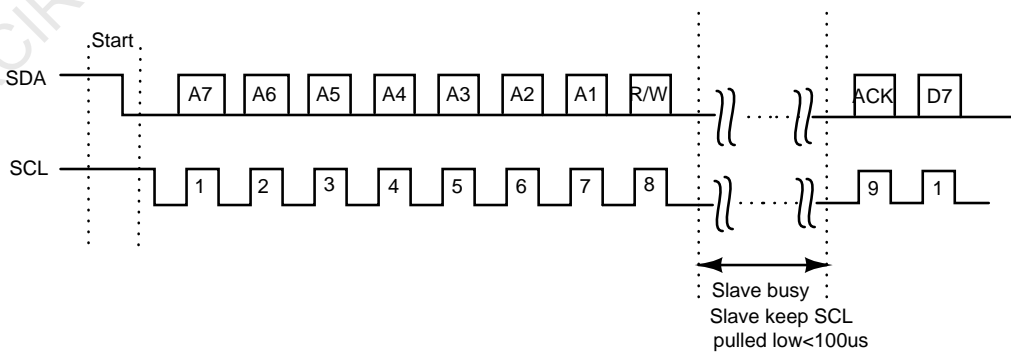


Figure 36: I²C clock stretching example

2 Data Protocol

The communication follows I²C convention. Refer to figure 37 for a definition of the symbols used.

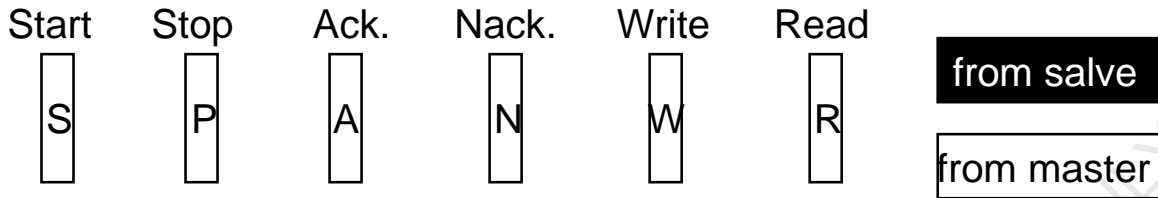


Figure 37: I²C symbols

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 13, table 15, table 15, table 16. And, the firmware in the slave will use a memory copy of the register for I²C slave read operation, so that firmware can continue updates, and I²C 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 39.
- 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 I²C read operation until the host finish the read operation. referred to first part of figure 40.
- I²C stop condition will release data protection and allow the slave firmware update the coordinates registers for I²C 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 I²C stop condition in each multi-fingers coordinates position reading, it will give the slave firmware chance to update the coordinates registers for I²C read operation, the host will give a combines unrelated data (combines new and old coordinates together), referred to the second part of figure 40.



Figure 38: Read operation

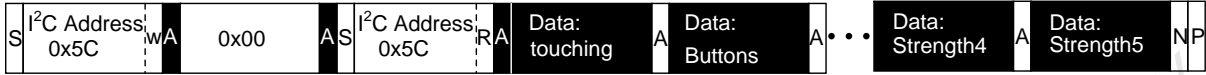


Figure 39: Coordinates read operation

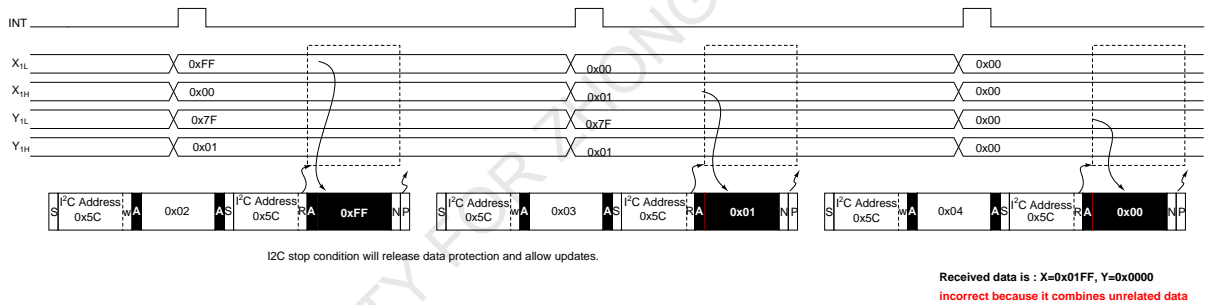
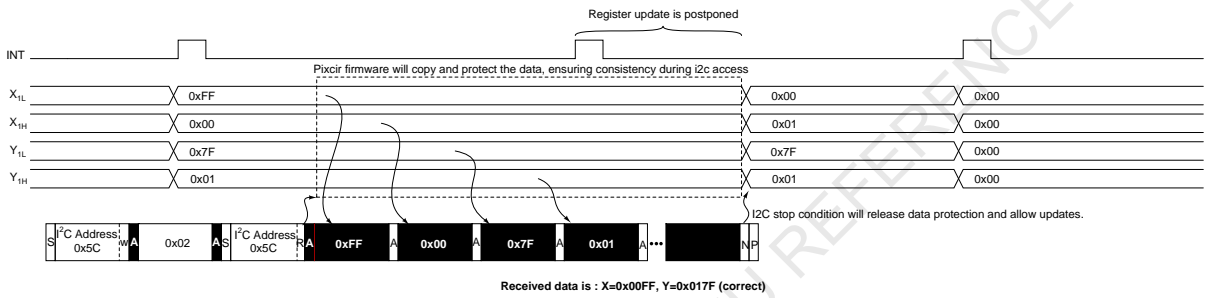


Figure 40: Coordinates read operation explanation

2.1.2 Write operation

NOT IMPLEMENTED

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 13, table 15, table 15, table 16. 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.

PIXCIR PROPERTY FOR ZHONGFU REFERENCE ONLY

dec@	Type	Name	Description	Category
0	char	touching	Bit field, see table 14	Touch Report
1	char	buttons	Buttons bitfield (1 bit per button), 1= button pressed	
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 Identifier code	
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 identifier	
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 identifier	
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 identifier	
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 identifier	
27	char	strength1	Finger #1 strength (X & Y merged) NOT IMPLEMENTED	
28	char	strength2	Finger #2 strength (X & Y merged) NOT IMPLEMENTED	
29	char	strength3	Finger #3 strength (X & Y merged) NOT IMPLEMENTED	
30	char	strength4	Finger #4 strength (X & Y merged) NOT IMPLEMENTED	
31	char	strength5	Finger #5 strength (X & Y merged) NOT IMPLEMENTED	

Table 13: MSI registers, Touch Report

Bit 0,1,2	Nb of fingers touching
Bit 3	noise flag (indicated that the level of noise detected is high) NOT IMPLEMENTED
Bit 4	bending flag (indicated that the level of bending is high) NOT IMPLEMENTED
Bit 5	buffer flag (indicates the master has missed more than 2 reports, which are stored in buffer array) NOT IMPLEMENTED
Bit 6	palm flag (indicates the algorithm has a palm or similar blocking issue)
Bit 7	water flag (indicates the algorithm has a rejected inputs due to water)

Table 14: Touching Bitfield

dec@	Type	Name	Description	Category
32 (lsb) 33 (msb)	int	Initial_distance	Distance separating fingers on the first time multitouch is detected NOT IMPLEMENTED	"Zoom" Gesture
34 (lsb) 35 (msb)	int	Distance	Distance separating fingers NOT IMPLEMENTED	
36 (lsb) 37 (msb)	int	Ratio	100-distance / initial_distance NOT IMPLEMENTED	
38	char	Water_level	NOT IMPLEMENTED	Monitor
39	char	Noise_level	NOT IMPLEMENTED	
40	char	Palm_level	NOT IMPLEMENTED	
41	char	Signal_X	Total signal strength X axis NOT IMPLEMENTED	
42	char	Signal_Y	Total signal strength Y axis NOT IMPLEMENTED	
43 50	char	Button1 .. Button8	Signal level of the buttons	Buttons
51	char	Power_mode	Power management register. See subsection §2.2.4 and table 21	Configuration
52	char	INT_mode	Control of the ATTB pin, see subsection §2.2.5 and table 22	
53	char	INT_width	ATTB pulse width	
54	char	Idle_freq	Scanning frequency in Sleep mode	
55	char	Auto_idle_delay	The delay time, the start is the last touch released in Active mode and the end is switch into Sleep mode successful	
56	char	Water_mode	Water mode settings	
57	char	Reserved	reserved for future use	
58	char	SPECOP	Special operation . See table 17	Special Operations
59 (lsb) 60 (msb)	int	SPECOP_IN_OUT_Data	Address used during special operation	
61	char	Engineering_cmd	Allows, with I ² c, to send "hyper terminal like commands" for engineering modes	Version
62 (lsb) 63 (msb)	int	CRC	FLASH CRC value (must be requested by SPECOP), excluding "EEPROM" zone	
64-113	char	version[0..49]	Customer version control (50 bytes linked to "EEPROM" Flash section)	

Table 15: MSI registers, gestures and monitoring

dec@	Type	Name	Description	Category
114-135	char	MESSAGE[0..391]	Null terminated ASCII message string for engineering and debug purpose. RESERVED - UNUSED NOT IMPLEMENTED	MESSAGE
136 (lsb) 137 (msb)	int	RAW_CTRL	Controls RAW data mode (internal, raw, etc. . .) see table 18	
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 . . . 77] (shared Buffer)	Raw data(X data first, Y data follow), content controlled by RAW_CTRL register, or alternatively, history buffer (see below)	RAW data

Table 16: MSI registers, raw_ctrl and data buffer

0x00	Normal operation
0x01	"EEPROM" read operation, start address must be written in SPECOP_IN_OUT_Data
0x02	"EEPROM" write operation NOT IMPLEMENTED
0x03	Calibration
0x04	Clear the auto-calibration offsets in RAM (only keep the EEPROM offsets)
0x05	Request a new auto-calibration in RAM
0x06	Soft reset (avoids the bootloader 60ms overhead)
0x07	CRC Check for all Flash Area (BOOT+APP+EEPROM). Result is written in "SPECOP_IN_OUT_Data"

Table 17: SPECOP

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.

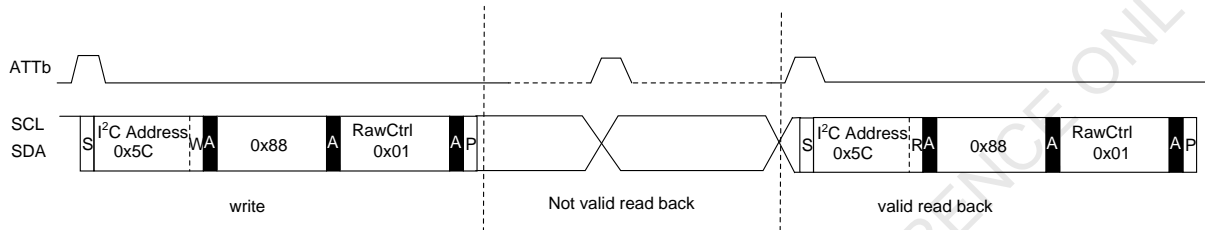


Figure 41: RAW_CTRL write&read

Bit 0	0: history buffer (not implemented), 1: RAW data, 2: System info, 23: Project info
Bit 1	info
Bit 2	Method (0 or 1), show RAW data with no correction at all
Bit 3	M0/M1: Show the EEPROM offset correction (only bits 0 and 3 must be set for M0)
Bit 4	M0: Also show (if enabled) the low-pass filter (only bits 0 and 4 must be set)
Bit 5	M1: Choose pattern small (0) or pattern large (1)
Bit 6	M1: Choose sense direction (0:Y, 1:X)
Bit 7	M1: Band scan, 0: only report a single cross node, 1 : report a full X axis scan at RAW position
Bit 8	Algorithm (0: ON, 1: OFF)
Bit 9	M0/M1: Enable single shot RAW refresh, must be set to 1 and bit10 to 0. Auto back to 0 and bit10 to 1 after single shot is done
Bit 10	M0/M1: Refresh frozen after single shot is done when 1. Set to 0 to release the freeze and go back to normal refreshing
Bit 11	M0: Also show (if enabled) the auto-calibration (only bits 0 and 11 must be set)
Bit 12	M0: Also show (if enabled) the sensitivity adjustment and the bending compensation (only bits 0 and 12 must be set)
Bit 13	M0: Also show (if enabled) the drift compensation (only bits 0 and 13 must be set)
Bit 14	Show Pen Raw data (0: Fingers, 1:Pen)
Bit 15	-

Table 18: RAW_CTRL

dec@	Type	Name	Description	Category
142	char	X_nbr	Number of X electrodes	System Info
143	char	Y_nbr	Number of Y electrodes	System Info
144	char	EEPROM_structure_version	HID Information	System
145	int	vid	HID Information	System Info
147	int	pid	HID Information	System Info
149	int	release	HID Information	System Info
151	int	X_res	HID Information	System Info
153	int	Y_res	HID Information	System Info
155	int	X_phys	HID Information	System Info
157	int	Y_phys	HID Information	System Info
159	int	CRC_boot	-	System Info
161	int	CRC_APP_FW	-	System Info
163	int	CRC_EEPROM_CONFIG	-	System Info
165	int	CRC_EEPROM_OFFSET	-	System Info
167	int	CRC_EEPROM_PATTERN	-	System Info

Table 19: MSI registers, system information

dec@	Type	Name	Description	Category
142	Int	TUNE_FW_version	GENERAL	Project Info
144	Char	FAE_name	GENERAL	Project Info
154	Char	date	GENERAL	Project Info
162	Char	customer_name	GENERAL	Project Info
173	Char	TP_module_maker	GENERAL	Project Info
181	Char	sensor_maker	GENERAL	Project Info
189	Char	customer_project_nickname	GENERAL	Project Info
199	Char	product_property	GENERAL	Project Info
200	Char	OS_name	GENERAL	Project Info
201	Char	interface	GENERAL	Project Info
202	Int	IC_solution	GENERAL	Project Info
204	Char	chip_location	GENERAL	Project Info
205	Char	TP_structure	GENERAL	Project Info
206	Char	pattern_type	GENERAL	Project Info
207	Char	cover_lens_material	GENERAL	Project Info
208	Char	cover_length_thickness	GENERAL	Project Info
209	Char	substrate_material	GENERAL	Project Info
210	Char	substrate_thickness	GENERAL	Project Info
211	Int	TP_size	GENERAL	Project Info
213	Char	ITO_pitch_size_x	GENERAL	Project Info
214	Char	ITO_pitch_size_y	GENERAL	Project Info
215	Char	fingers_n_report	GENERAL	Project Info
216	Char	pen	GENERAL	Project Info
217	Char	X_nbr	GENERAL	Project Info
218	Char	Y_nbr	GENERAL	Project Info
219	Char	Button_nbr	GENERAL	Project Info

Table 20: MSI registers, Project_Info

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 21

Bit	Name	Description
7-3	-	Not used
2	ALLOW_IDLE	Allow self demotion from active to idle 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 idle mode. If this flag is not set, the host must explicitly switch the device from active to idle mode.
1-0	POWER_MODE[1-0]	Power mode setting of the MSI device: 00: Active Mode 01: Idle Mode 11: Sleep Mode

Table 21: Power_mode register

2.2.5 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

Table 22: INT_mode register

2.3 Details about coordinates report and finger ID

2.3.1 Double buffering and atomic access

The I²C 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 I²C table are refreshed. Concurrently, the I²C master may consult the I²C registers table and read one byte after one byte the coordinates, a process whose speed is limited by the I²C baud rate. It is important that the action of refresh, performed after the sensor scan, does not collide with the access by the I²C master. Otherwise, the I²C 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 I²C 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 I²C 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 I²C bus is released, an instant buffer swap can be performed, presenting the freshest set of coordinates.

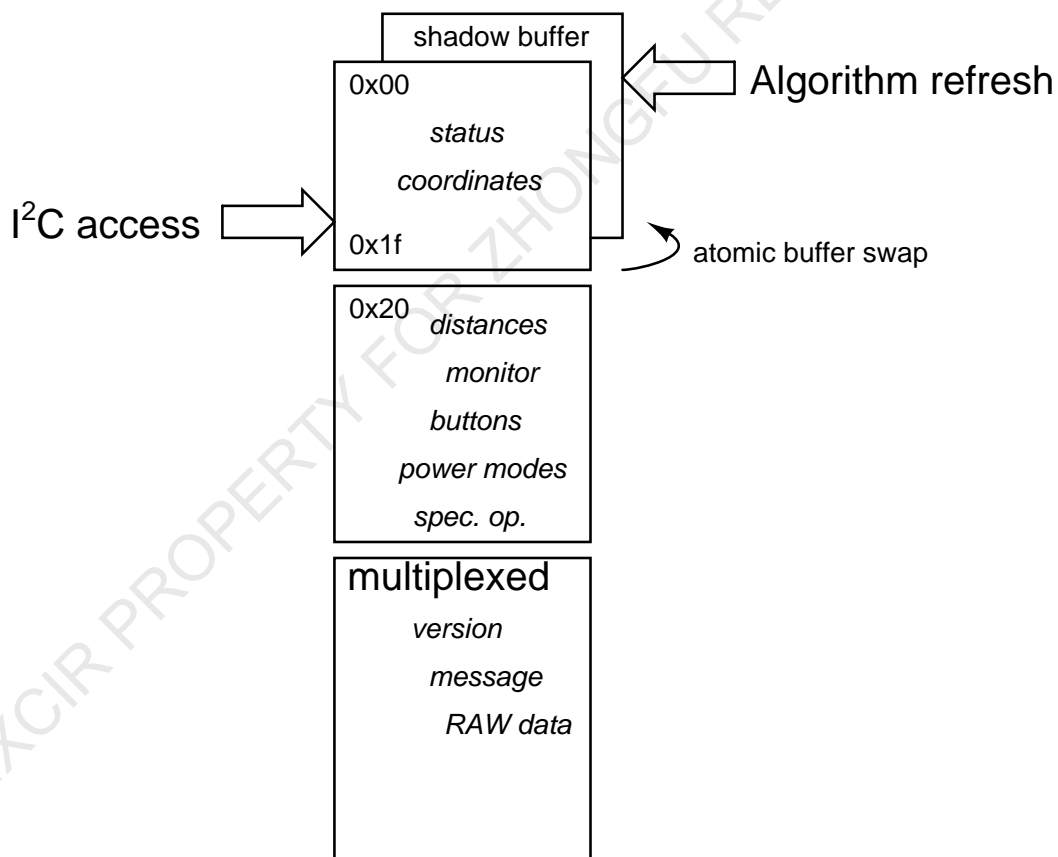


Figure 42: Registers table overview

The conditions for the mechanism to work are the following:

- **The read operation of the coordinates table must be performed with a single I²C 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 I²C buffer, and the master would retrieve a mix of data. One scan time is approximately 4ms, therefore the I²C read must finish within approximately 8ms. Reading 32 bytes with an I²C speed of 100bit/s takes normally less than 3ms.
- It is mandatory to release the I²C 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 23. One important value stored in TOUCH register is NBF (number of fingers)

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

Table 23: TOUCH register

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 I²C 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 I²C 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 I²C data. On the other hand, the order of the slots used for touching finger can not always be guaranteed. 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 I²C 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.

example: 3 touching fingers (NBF=3)

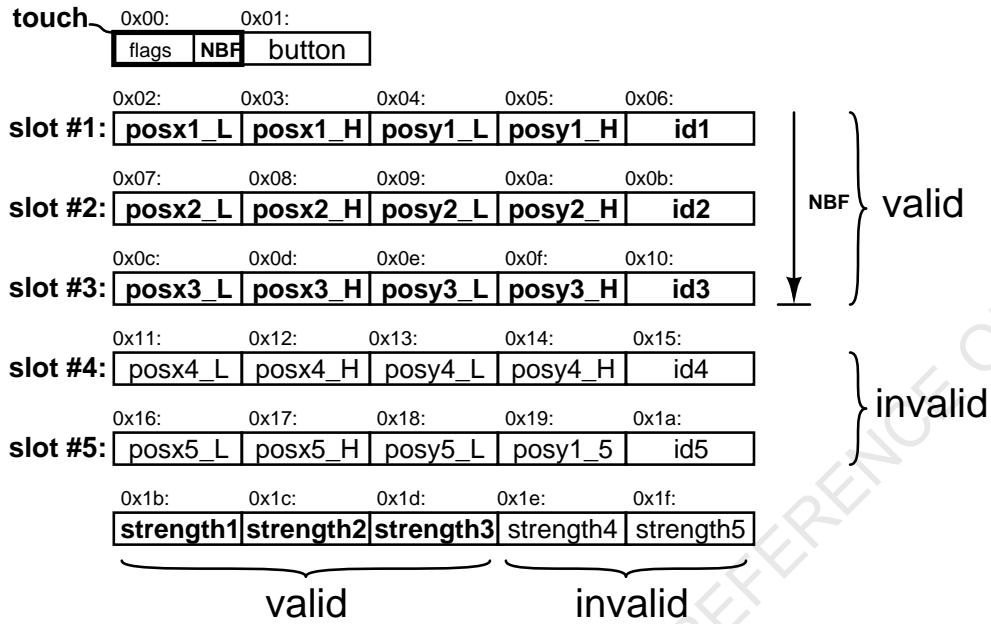


Figure 43: Coordinates section

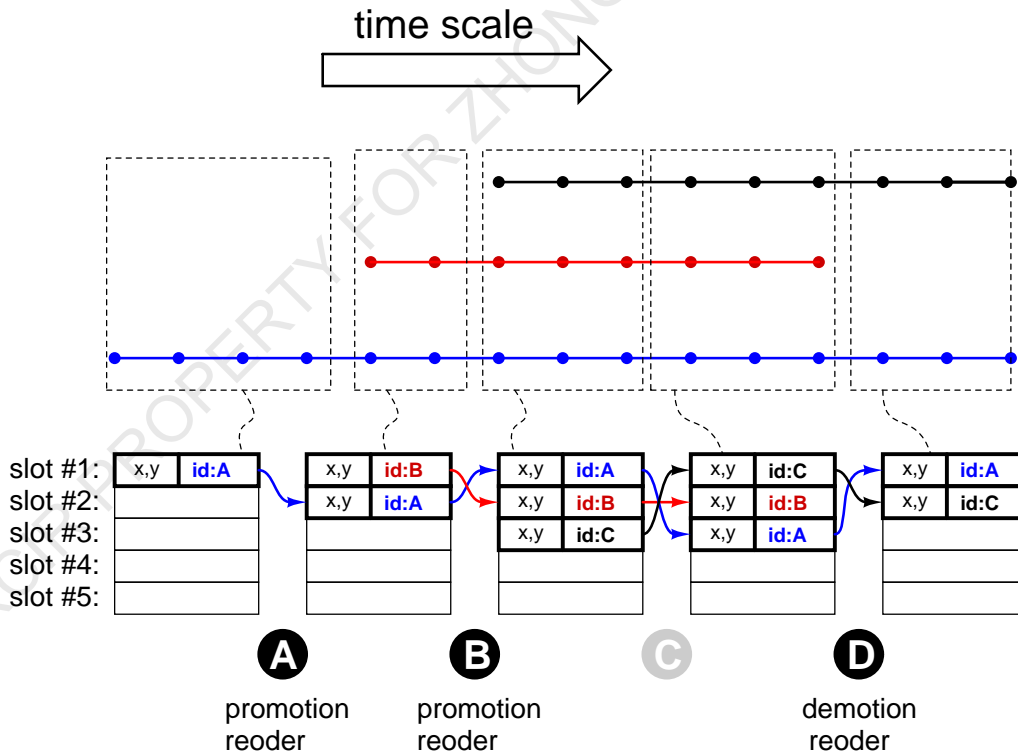


Figure 44: Finger slot system. Slot order is never guaranteed, the master must always check the finger ID tag. Empty slots are however always at the end of the i²c table.

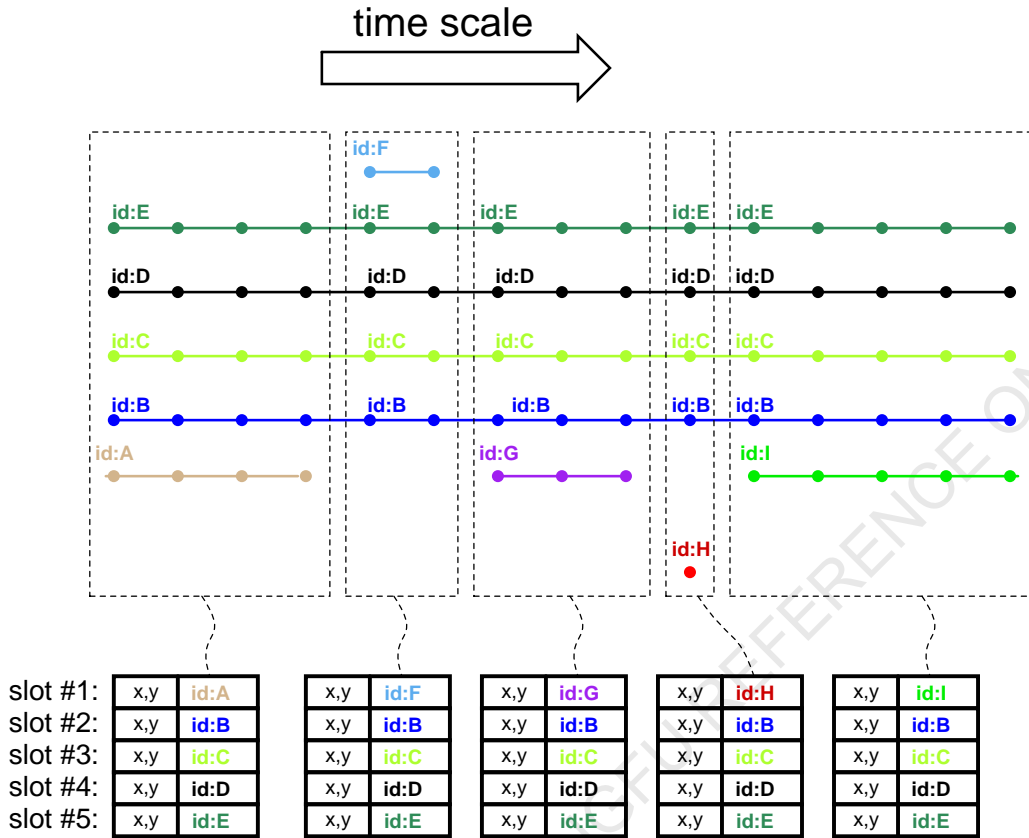


Figure 45: Finger ID system. Dots which can not link to an older report get a new finger ID assignment.

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 I²C 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.

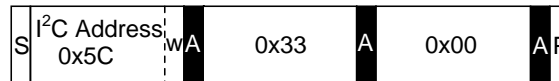


Figure 46: Active mode sequence

2.4.2 Idle 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 controlled by **Idle_freq** in table 15). 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 Idle mode when no finger is detected for more than IDLE_PERIOD time, provided that ALLOW_Idle bit is set in the POWER_MODE register. Figure 47 shows how to force the slave into Idle mode. Figure 48 shows how to force the slave to switch automatically into Idle mode (set ALLOW_IDLE bit in POWER_MODE register).

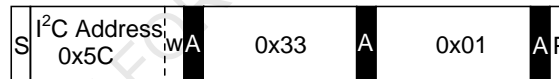


Figure 47: Idle mode sequence

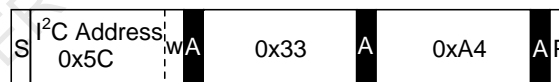


Figure 48: Idle mode automatically switch sequence

2.4.3 Sleep 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 Sleep mode. Here is **only one way** to wake up from sleep mode.

- RST pin pull up 10 ms (connect to Vcc)

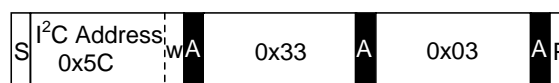


Figure 49: Sleep mode sequence

2.4.4 Power consumption

Type measure: hardware: C44+F32, 10.1 inch ITO(39X 22Y); FW version MP6

Power Supply			3.3V			
Power Mode	Fingers	$F_{scan} (Hz)$	Min.	Typ.	Max.	Unit
Active mode	1	180*	-	3.96	4	mA
	2	160	-	4.76	5	mA
	3	122	-	4.88	5.2	mA
	4	92	-	5.13	5.4	mA
	5	75	-	5.26	5.6	mA
Idle mode	0	10	-	0.09	0.11	mA
Sleep mode	-	-	-	2	2	uA
bootload	-	-	-	5.79	6.2	mA
Calibration	-	-	-	6.12	6.2	mA

Table 24: Power Consumption list

Depend on ITO size, parameters set and noise level are different, report rate will be difference.
*2 fingers special version FW one finger report rate can get 280Hz.

The figure 50 shows that how is the I²C wake up from idle mode and it shows its Power Consumption .

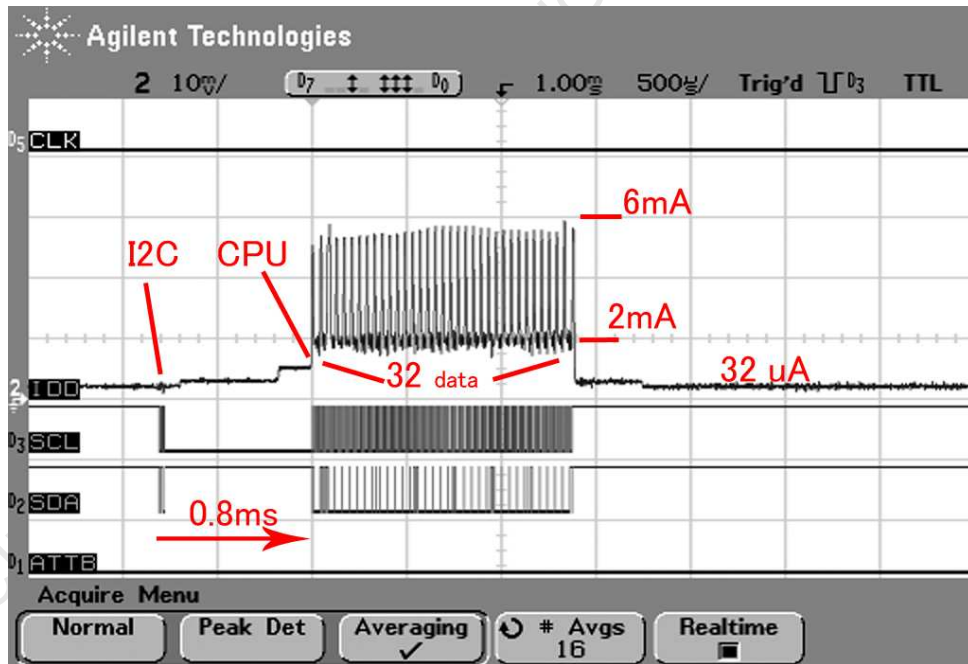


Figure 50: The example for I²C wake up

2.4.5 Power mode diagram

Figure 51 shows the power mode change diagram.

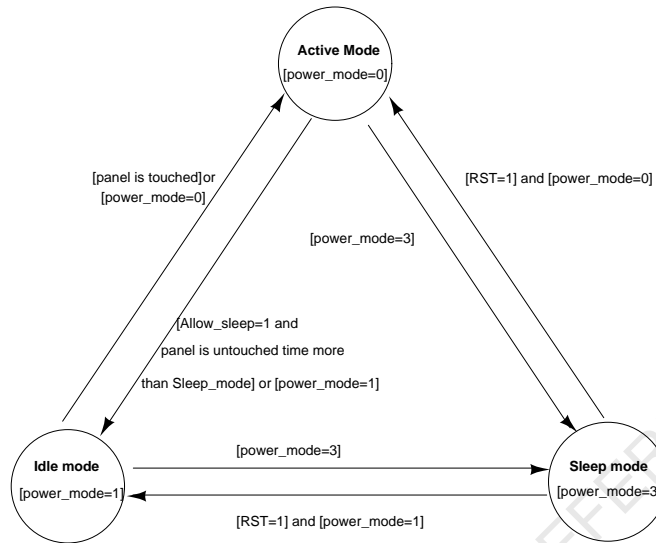


Figure 51: Power mode diagram sequence

Switch	Time
power up time	same as recovery time
recovery time	73.85ms(include RST pulse 10ms)
active to idle	1.35ms(IIC command)
active to sleep	1.35ms(IIC command)
idle to active	95.50ms
sleep to active	100.54ms(include RST pulse 10ms)

Table 25: Power mode switch time

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 `SPECOP_IN_OUT_Data`.

2.5.3 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(manual 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 the `SPECOP` register will take back its default zero value. The sequence for calibration is shown in the figure 52. 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 $T_{CAL} < 500$ ms (for one Tango solution). The `ATTB` line will be pulled up after finish calibration. In really case depend on panel size, SPI speed and noise level... will cost different time. figure 53 (C44+F32;X axis 39;Y axis 22)



Figure 52: Calibration sequence



Figure 53: C44+F32;X axis 39;Y axis 22

2.5.4 Border tracking calibration

Drift Analysis

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

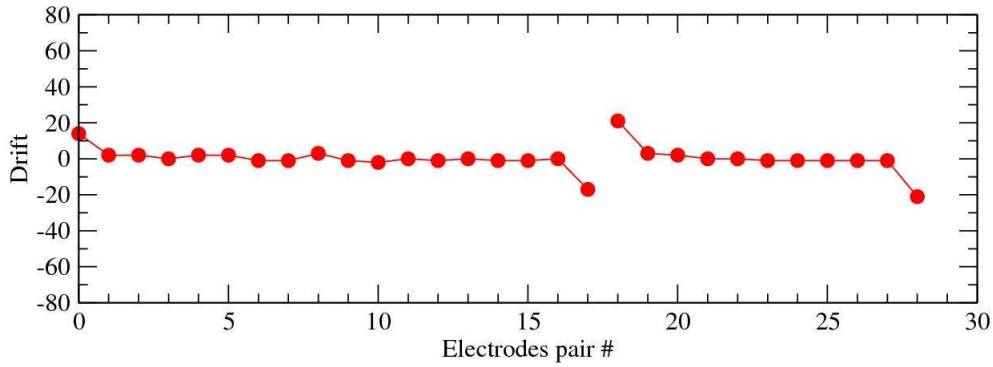


Figure 54: Storage test

High humidity test (drifting during the use):

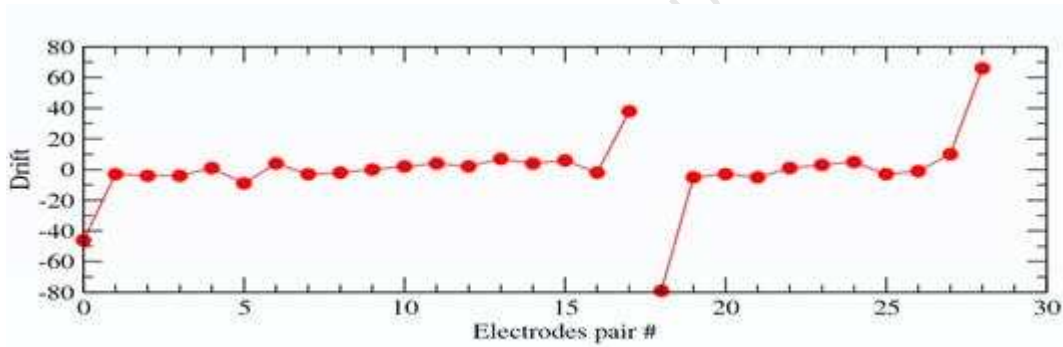


Figure 55: High humidity test

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

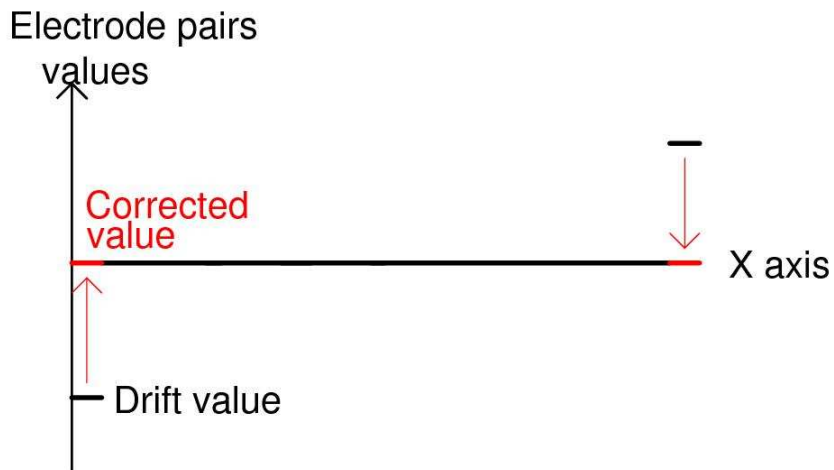


Figure 56: Border electrodes drift value

- 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 0x07, the FW will compute CRC checksum operation for Bootloader area, APP FW area, EEPROM Config area, EEPROM offset Area and EEPROM pattern area. The result will be wrote and accessible in the SPECOP_IN_OUT_Data register. result is a bit field:

- Bit0: BOOT CRC result (0: Wrong CRC - 1: CRC OK)
- Bit1: APP FW CRC result (0: Wrong CRC - 1: CRC OK)
- Bit2: EEPROM config CRC result (0: Wrong CRC - 1: CRC OK)
- Bit3: EEPROM offset CRC result (0: Wrong CRC - 1: CRC OK)
- Bit4: EEPROM pattern CRC result (0: Wrong CRC - 1: CRC OK)
- Bit5->Bit13: unused bit14&15: CRC checksum result status (0: Nothing - 1:Failed - 2:Pending - 3:Ready)

Note : bit14&15 must be equal to 3 to assure result data reliability

2.5.6 Soft reset (jump to bootloader)

When SPECOP is written with the code 0x06, 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 Auto calibration

This section explain the auto calibration.

2.6.1 Introduction

EEP_DRIFT_COMP = 0, all disable

EEP_DRIFT_COMP = 1, only drift compensation is enabled.

EEP_DRIFT_COMP = 2, only the auto calibration is enabled.

EEP_DRIFT_COMP = 3, both, drift compensation and the auto calibration are enabled.

2.6.2 Offset stored in flash

The auto calibration stores the offsets in RAM. But the initial offsets of the device must be stored in flash. 2 ways to store the initial offsets:

- Record the offsets and write it in the XML file. The "EEPROM_make" program stores the offsets in the EEPROM.pix. It is mandatory.
- Send a calibration command (manual calibration). It is recommended.

2.6.3 When the offset must be stored in the flash ?

To store the offsets in the flash, the user must send a calibration command (manual calibration) or load a new eeprom with the appropriated offsets. **The offsets must be stored in the flash each time the Raw data of M0 did change because of mechanical assembly.** During mechanical assembly, raw data of M0 can change a lot. Figure 57 shows an example of mechanical assembly.

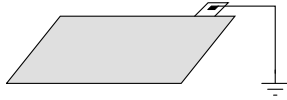
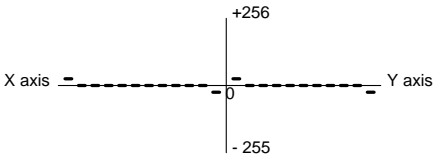
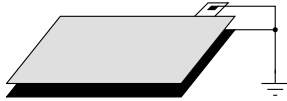
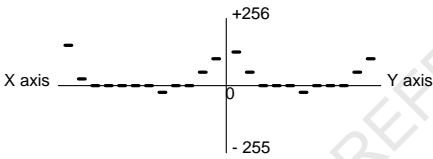
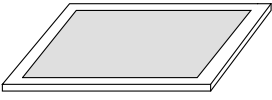
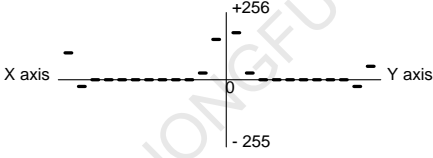
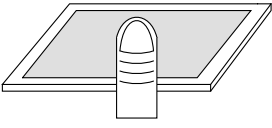
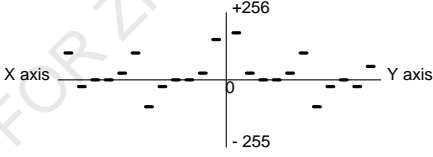
Assembly	M0 raw data	User action
ITO only 		Store offsets in the flash
ITO + LCM 		Store offsets in the flash
Device fully assembled 		Store offsets in the flash
finger touch 		Do not Store offsets in the flash

Figure 57: Mechanical assembly

Conditions to store the offsets in the flash:

- No finger or any conductive stuff must touch the TP.
- The offsets are stored in the flash for a dedicated assembly step. Test the TP in the same conditions as the offsets storage.
- Avoid source of noise, like AC power supply, LCM on, when you store the offsets in the flash.

2.6.4 What is the purpose of the auto calibration?

Auto calibration is a term not well understood by many agents and customers. If the assembly of the device is robust and a calibration command is done at the end of the assembly, there is no need to do auto calibration, the “drift compensation” would be enough. Any way, the auto calibration is working in different cases, depending of the environment. The Figure 58 shows a table with different cases.


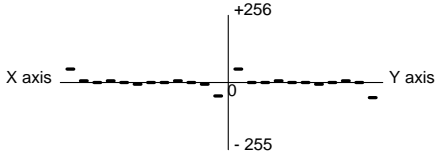

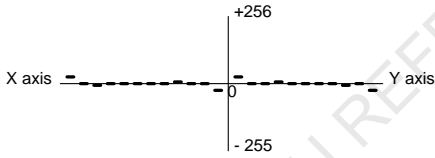

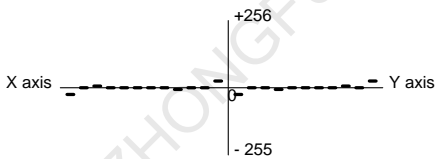
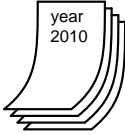
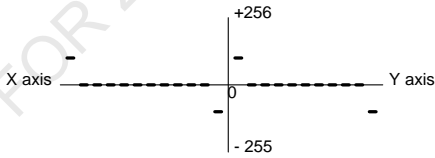
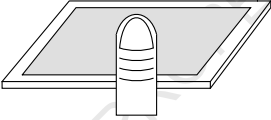
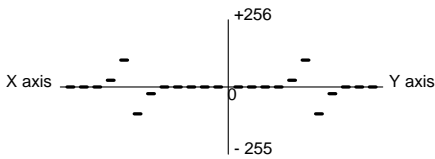
Environment/Cases	M0 raw data - offsets stored in the flash	Firmware action at power on
High humidity 		The firmware stores the offsets in RAM.
High temperature 		The firmware stores the offsets in RAM.
Low temperature 		The firmware stores the offsets in RAM.
Time 		The firmware compensates the drift in RAM.
Finger touching the TP 		The firmware does NOT save any offsets.

Figure 58: auto calibration purpose

2.6.5 Summary of the auto calibration

The limits and conditions of use are listed below:

1. The default offsets of the method 0 and method 1 data must still be stored in the flash first!
2. Auto calibration is used only for small variations of M0 raw data. These small variations (called offsets) are written in the RAM.
3. Auto calibration compensates the default offsets stored in the flash.

2.7 Autocalibration for buttons

To enable the feature autocalibration for buttons in the FW, several changes on HW and EEPROM configuration side are necessary:

1. Schematic change for the external Lift for button autocalibration
2. Additional scan procedure for the autocalibration for Buttons
3. Configuration of the button autocalibration EEPROM parameters

2.7.1 Schematic change for the external Lift for button autocalibration

The necessary changes in the schematic are shown in figure 59.

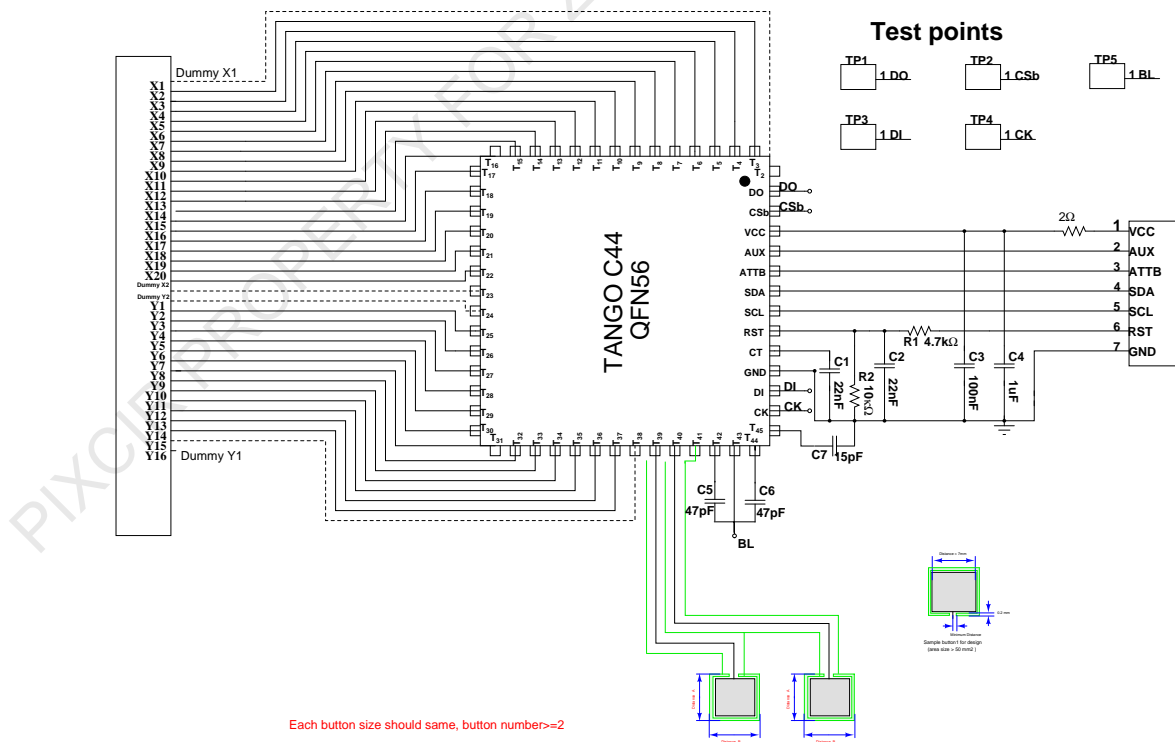


Figure 59: Schematic for C44 with simple button autocalibration

The extra components needed for the button autocalibration in the schema are C5, C6 and C7. The external capacitances C5 and C6 are used for the scan with external Lift and C7 is used for the Offset compensation.

Button dimension The effective area of the buttons should be $>50\text{mm}^2$. This rule should be also applied if the Button includes a hole in the middle. The value of C7 is related to the effective area of the button. With an effective area $>50\text{mm}^2$, the C7 value is approximately 15pF.

2.7.2 Scan procedure for Autocalibration Button

The autocalibration for button feature should perform the following scan procedures (see figure59 for the references of the pins) :

1. Scan with external Lift (single ended). For this scan the pins have to configured in the following way (repeat scan with each Button as sense):
 - a) T40 (or T39): Sense (button Sense)
 - b) T41: unused (compensation not needed)
 - c) T42: Sense (external Sense)
 - d) T43: Lift (external Lift)
 - e) T44: Ref (external Ref)
 - f) T45: Ref (compensation Ref)
2. Scan between buttons (differential scan) with additional external lift. For this scan the pins have to configured in the following way and repeat for each Button:
 - a) T39: Ref (button Ref)
 - b) T40: Sense (button Sense)
 - c) T41: Lift (normal Lift)
 - d) T42: Sense (external Sense)
 - e) T43: Lift (external Lift)
 - f) T44: Ref (external Ref)
 - g) T45: unused

2.7.3 Configuration of the button autocalibration EEPROM parameters

Enable the autocalibration for buttons in the EEPROM (EEP_BUTTON_AUTOCAL_ENABLE) and configure the EEPROM parameters (EEP_BUTTON_AUTOCAL_*) accordingly.

2.8 Coordinates characteristics

The reported posX and posY integer coordinates of the finger(s) position(s) are related to the indexes illustrated on figure 60, 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 $\text{posX} = 2.7 * 512 = 1382$ and $\text{posY} = 1.4 * 512 = 717$.

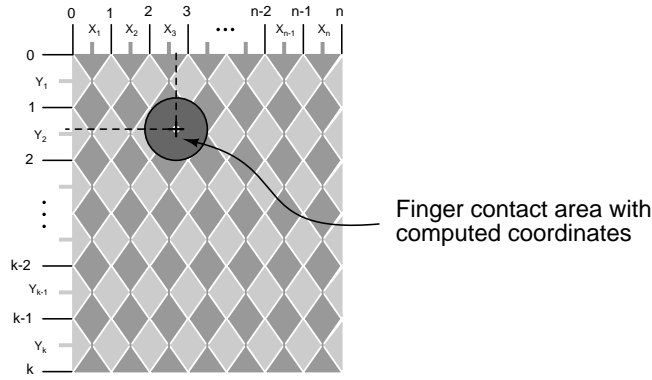


Figure 60: Active area coordinates indexes

As illustrated on figure 61, in the borders 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 ($\Delta x \neq 0$ in this example), whereas the coordinate of the other axis (Y in this example) is not affected ($\Delta y = 0$ 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 62.

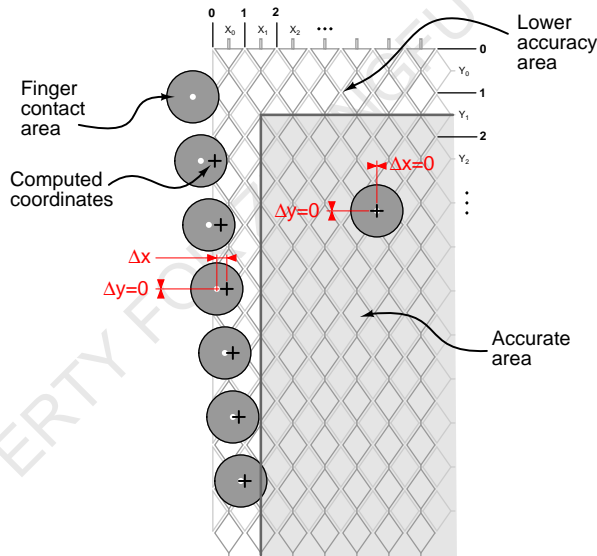


Figure 61: Accuracy of the coordinates vs real finger position on active area

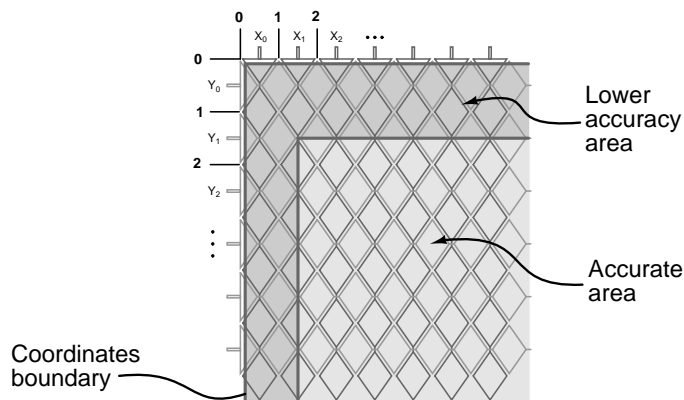


Figure 62: Active area accuracy zones and coordinates boundary

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 I²C address:
0x5C

Bootloader mode which allows loader operation. The device uses another I²C address: 0x5D,
and uses a specific syntax.

In bootloader mode, the device uses a different, more simple I²C servicing firmware, which is totally separated from normal I²C. This means that the entire firmware of normal mode can be re flashed, including the I²C code itself

3.1.1 Bootloader general flow diagram

The flow chart 63 illustrates the general process flow of the Bootloader FW.

PIXCIR PROPERTY FOR ZHONGFU REFERENCE ONLY

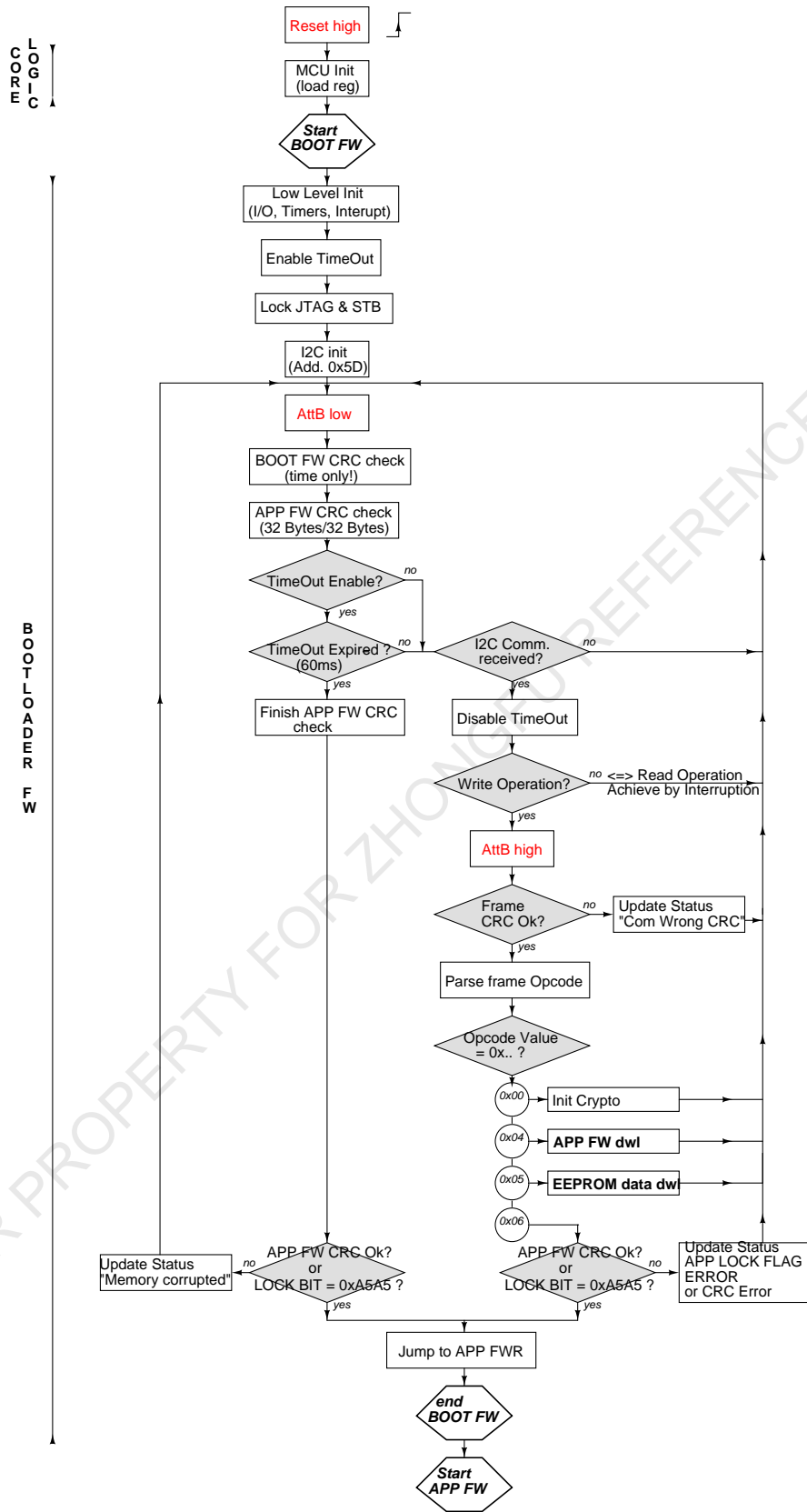


Figure 63: Bootloader general flow diagram (for illustration purpose only)

3.1.2 Bootloader Application FW download flow diagram

This flow chart 64 illustrates the process flow during Application FW download.

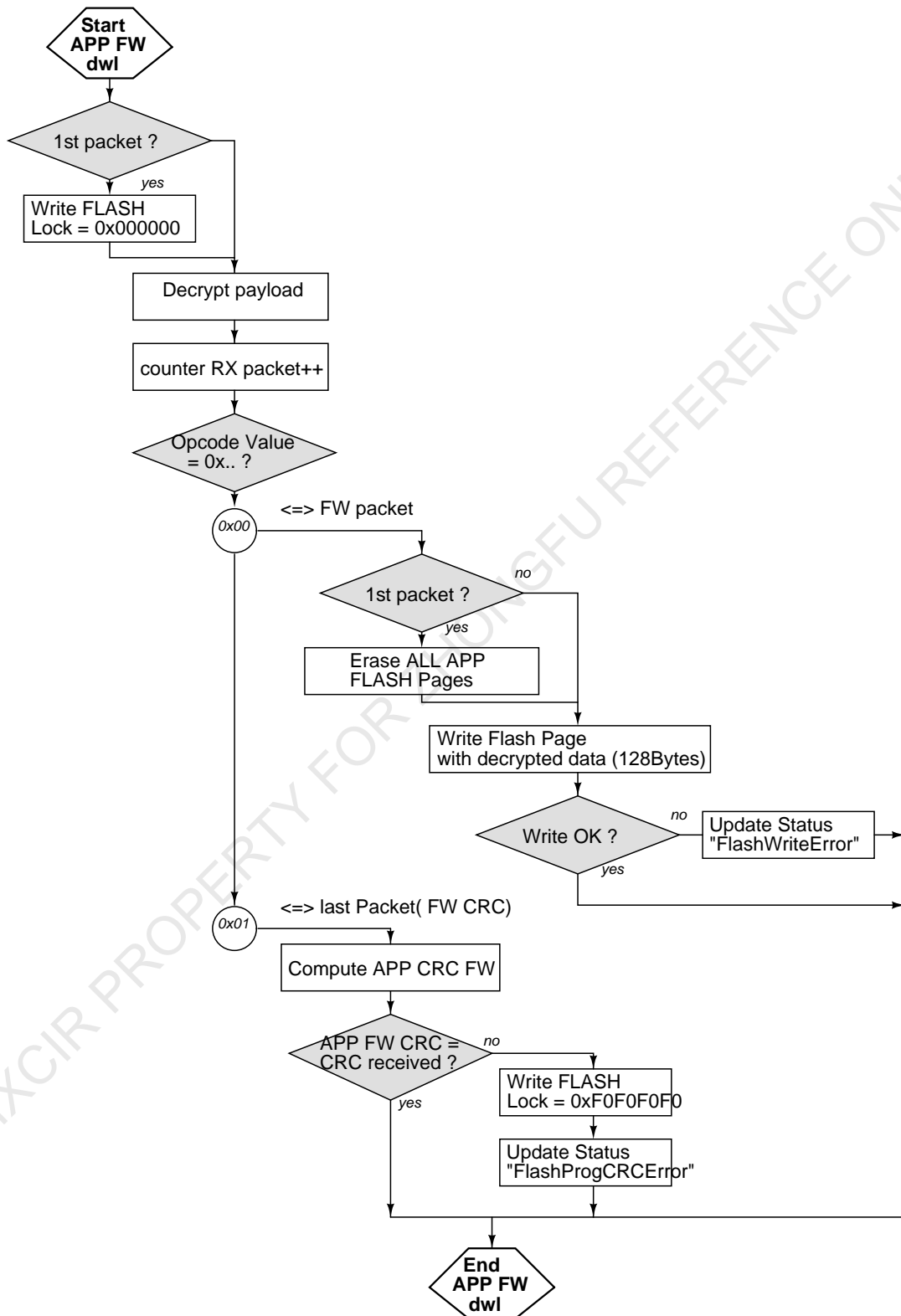


Figure 64: Bootloader “APP FW dwl” flow diagram (for illustration purpose only)

3.1.3 Bootloader EEPROM download flow diagram

This flow chart 65 illustrates the process flow during EEPROM download

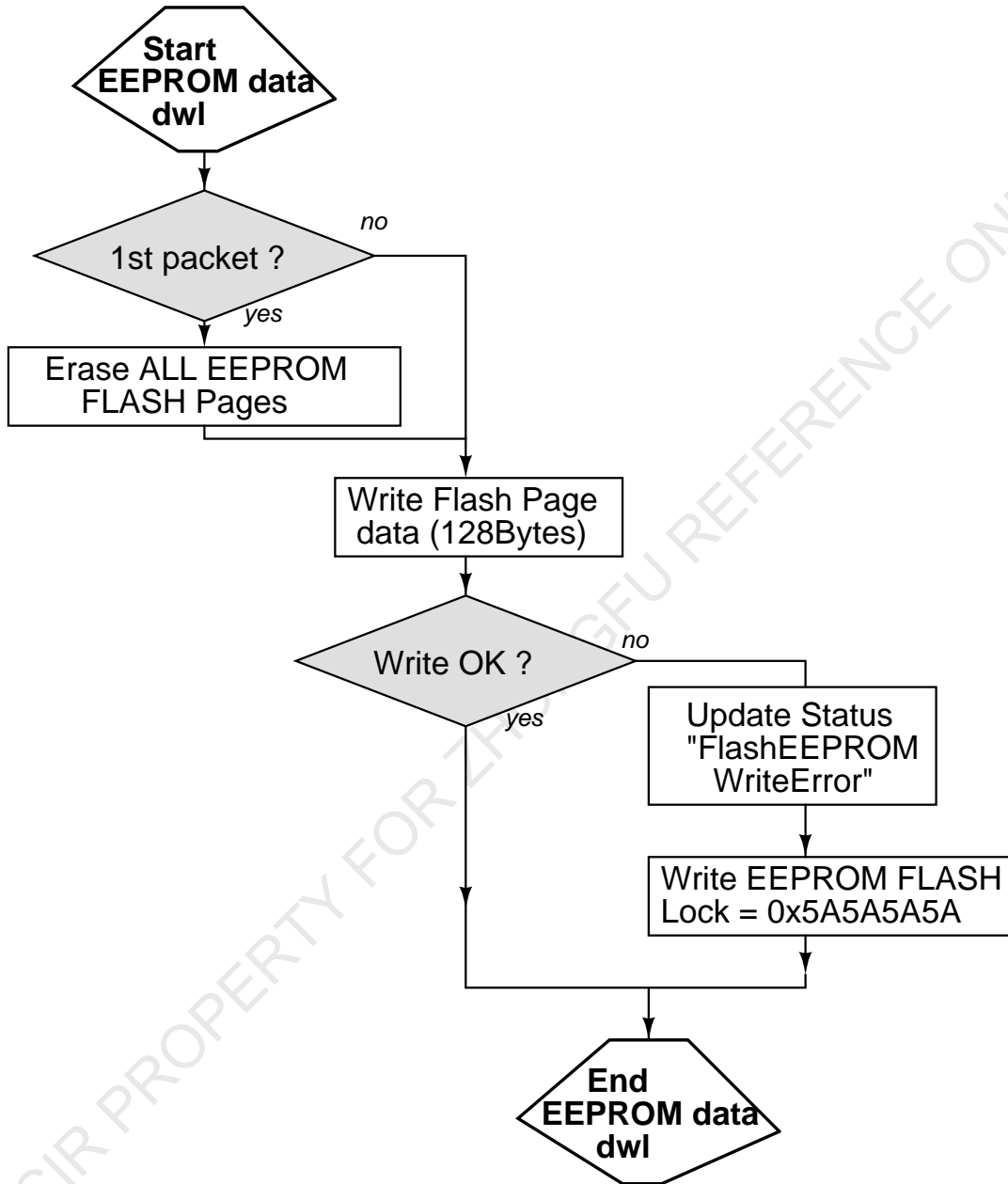


Figure 65: Bootloader “EEPROM data dwl” flow diagram (for illustration purpose only)

3.2 Switching to bootloader

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

- by software, by modifying the SPECOP register by an I²C command to address 0x5C.
- by hardware reset, followed by an appropriate I²C command to address 0x5D.

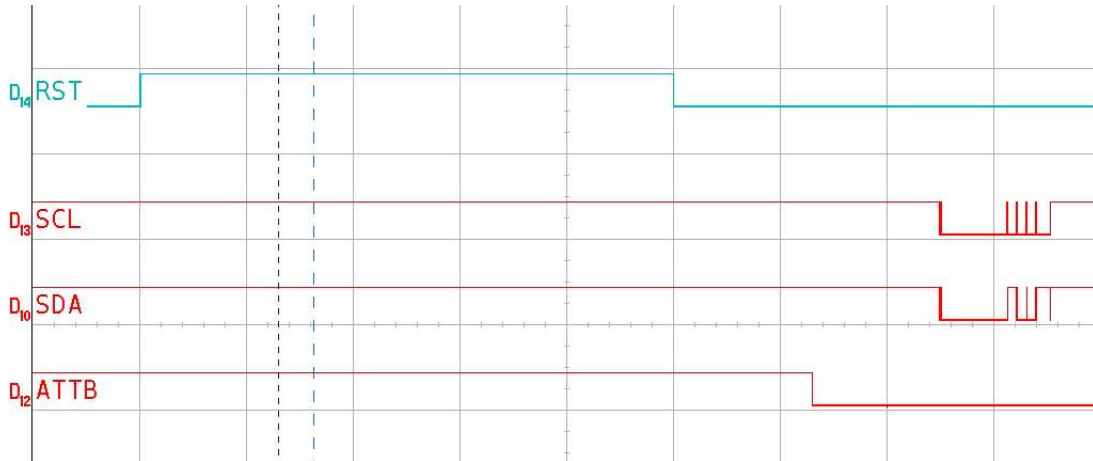


Figure 66: Timing sequence for switch to bootloader

3.3 Bootloader read operation

In bootloader mode, I²C 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, I²C 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 I²C transaction. The bootloader can understand the following OPCODE commands:

- INIT (code 00): reinitialize the cryptography engine, and allow for a new upgrade
- USER (code 06): switch to normal mode.
- AES (code 04): load a frame. Frames are decrypted and interpreted after the I²C STOP transition. The device will release the INT signal and will not respond to any I²C 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 05): allow user to modify EEPROM without cryptography. The CRC error flag is set if either transmission or writing to the EEPROM failed

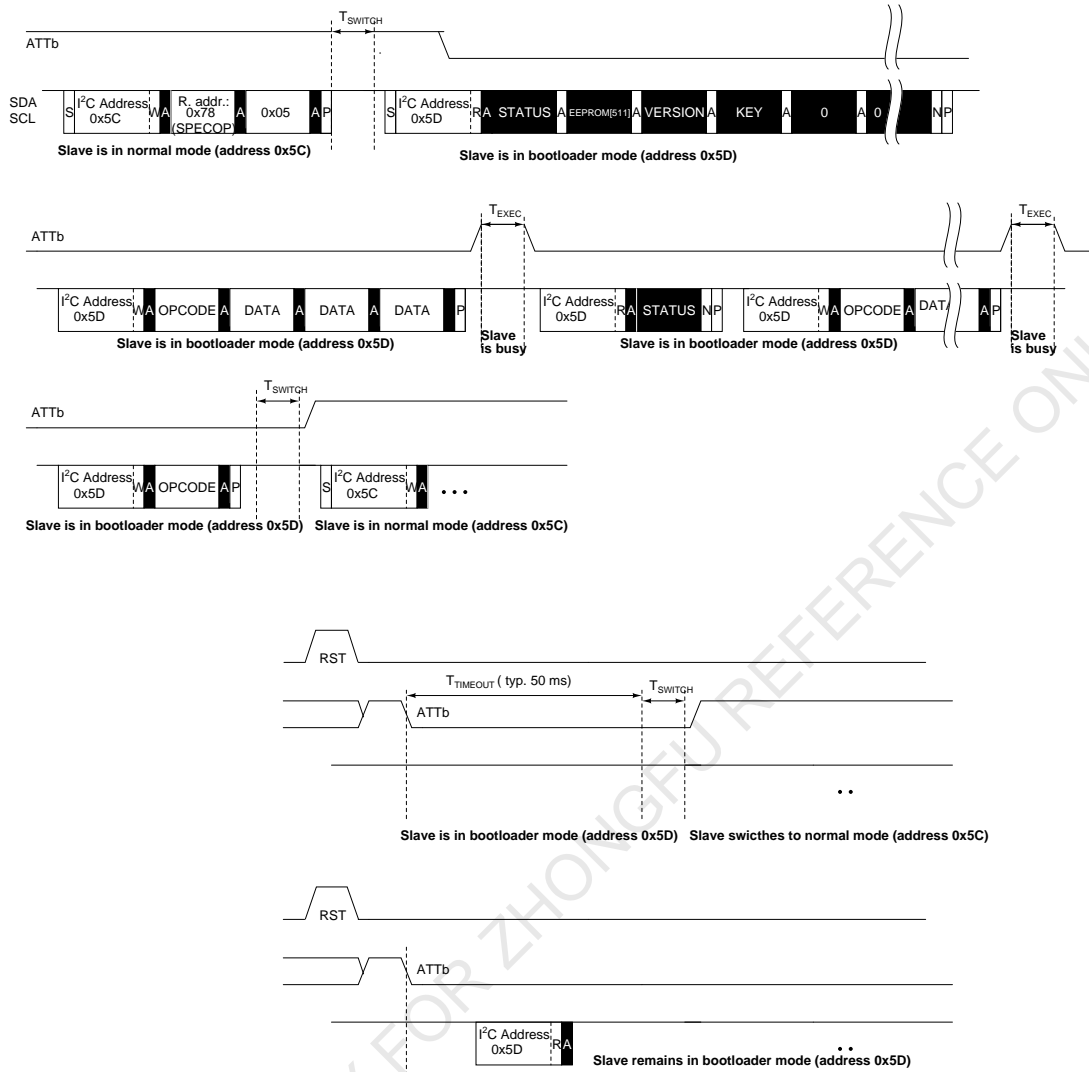


Figure 67: Examples of I²C communication. Top row shows the transition from normal to bootloader mode, followed by a read operation which reads the status, bootloader version and crypto key information. The row in the middle shows the sequence of sending data frames, followed by a flashing operation, 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 Frame format

The frame format is shown in figure 68.

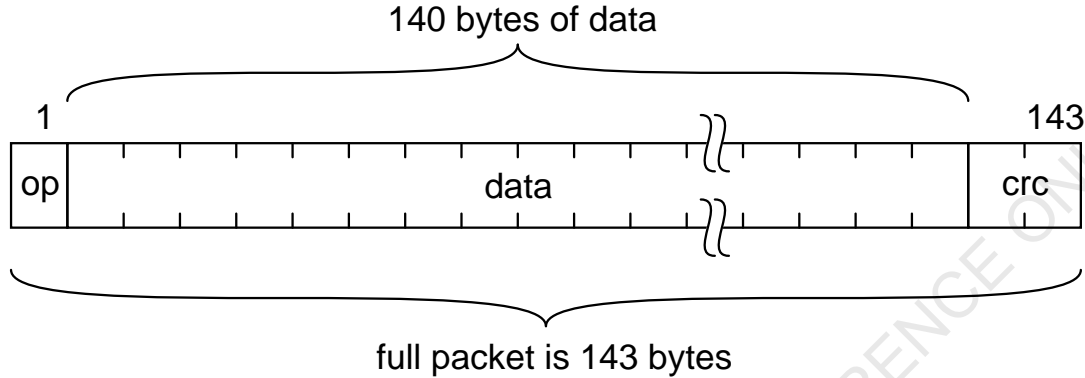


Figure 68: Format of an encrypted frame

The OPCODE (1 byte, code 04) 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.

3.5.2 File example

```
! Comment example
$043FD38AE5706C42BCDFD533DFD39D860C90FF9672AFC69E47CA035BE2775F
$049C081C92AD4FC9E74B000DOB669B0445E86DFC37824D0D7702D4BC6DBC34
! Comment again here
$04E3C7B35082AD22A61DFOEB571F758DFB61ED6D27964EE99B44AD5D31317D
```

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 26: 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 occur, the host should send the frame following the FRMCNT value. For example, if a CRC error occurs and if FRMCNT==3, the host should send against 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

PIXCIR PROPERTY FOR ZHONGGU REFERENCE ONLY

3.7 Example

Figure 69 shows bootloader behavior if it does not receive I2C commands (no FW download...)

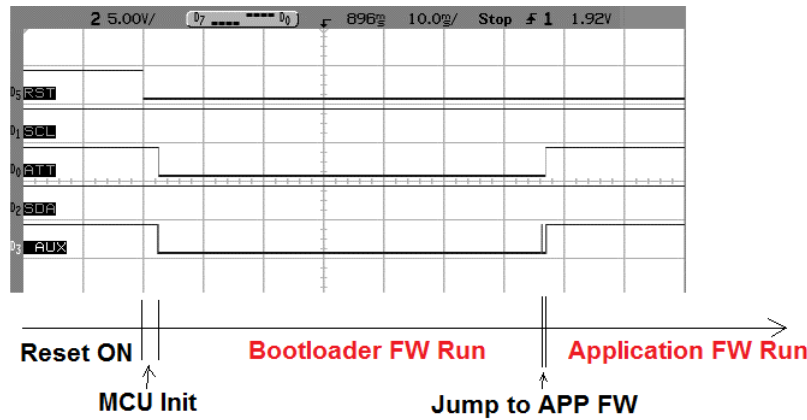


Figure 69: Bootloader “normal” execution

The figure 70 shows an Overview of a full firmware upgrade. Reset pulse is followed by the send of I2C command “Init Crypto” then the download of all new APP FW frames. When all is done the APP FW CRC computation is proceed before finally jumps into APP FW and start of scan.

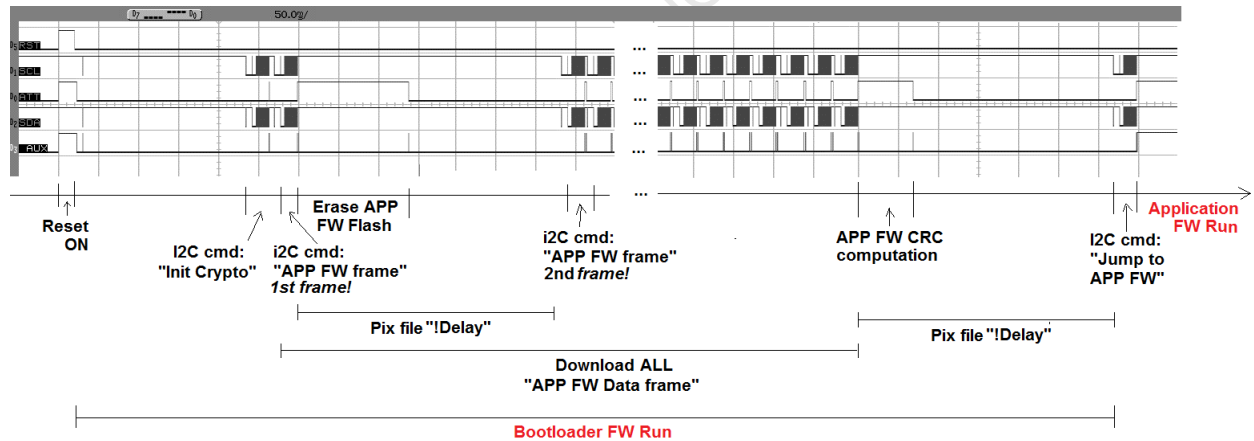


Figure 70: Firmware Download (PIX file FW)

4 EEPROM parameters

This chapter briefly describes the TangoC EEPROM structure. For details about the available EEPROM parameters please refer to Application Note 711: “TangoC EEPROM structure reference”.

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
{
    eeprom_t;
    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`:

eeprom_t Basics project information and power mode control

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

5 Packaging information

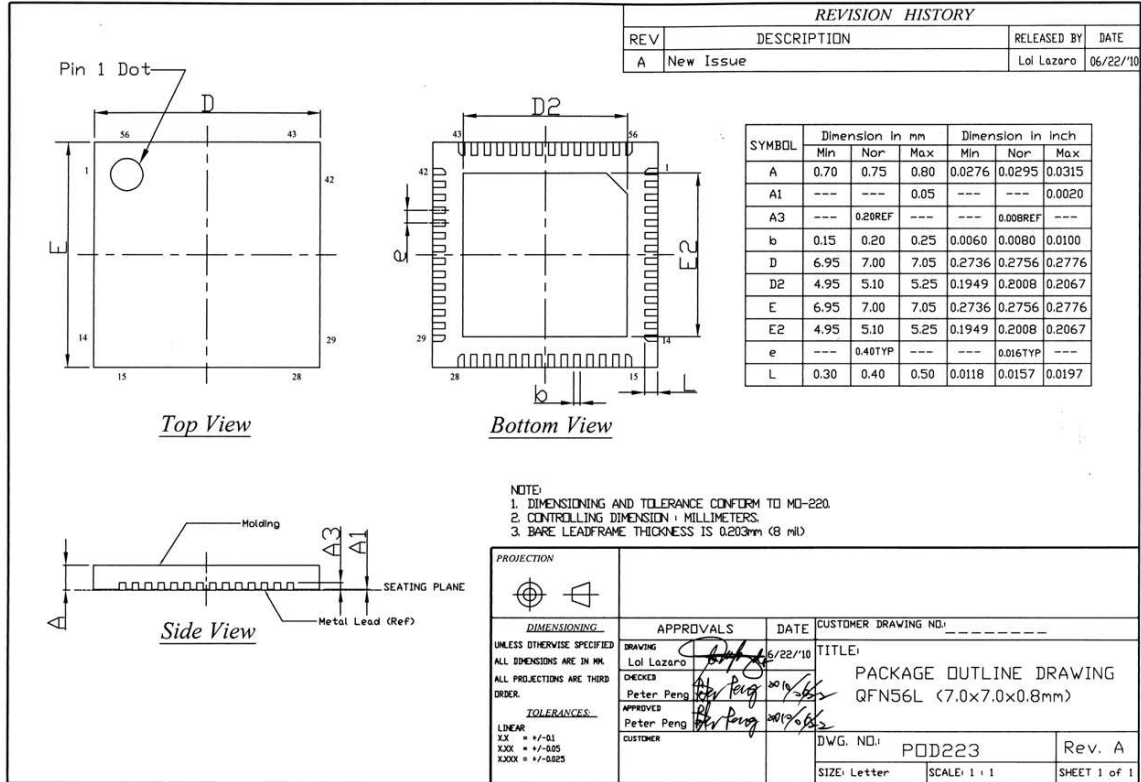


Figure 71: Tango C44/C48 Packaging information

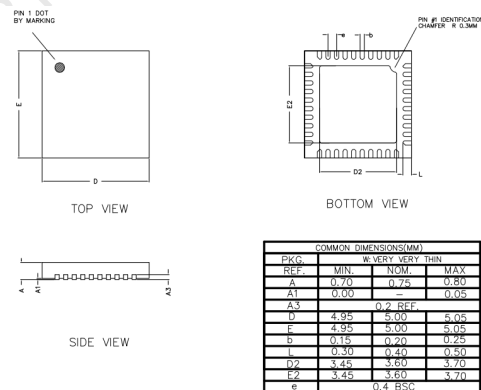


Figure 72: TANGO C32 Packaging information

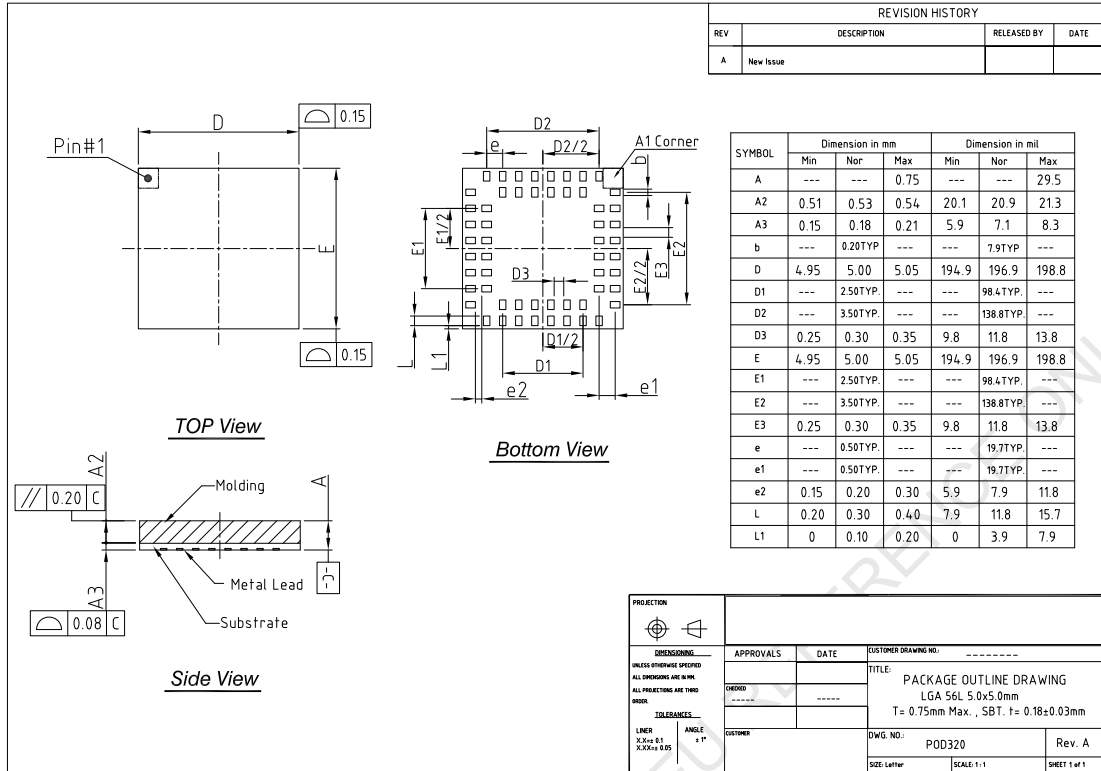


Figure 73: Tango C48S LGA56 Packaging information

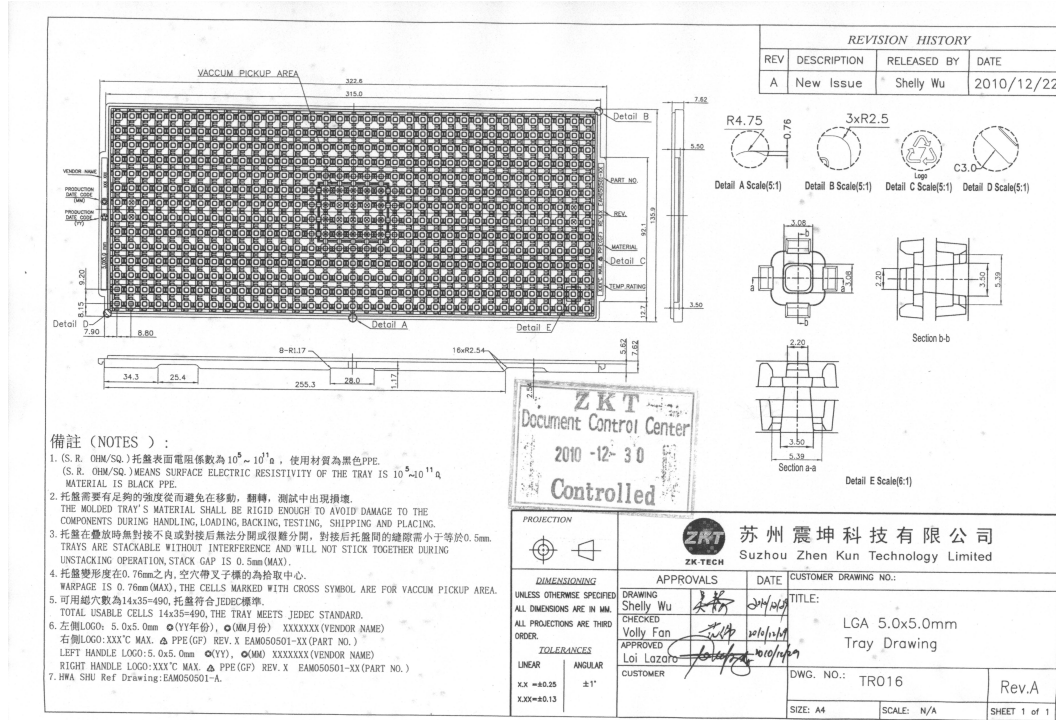


Figure 74: Tray C32

PIXCIR PROPERTY FOR ZHENKUN

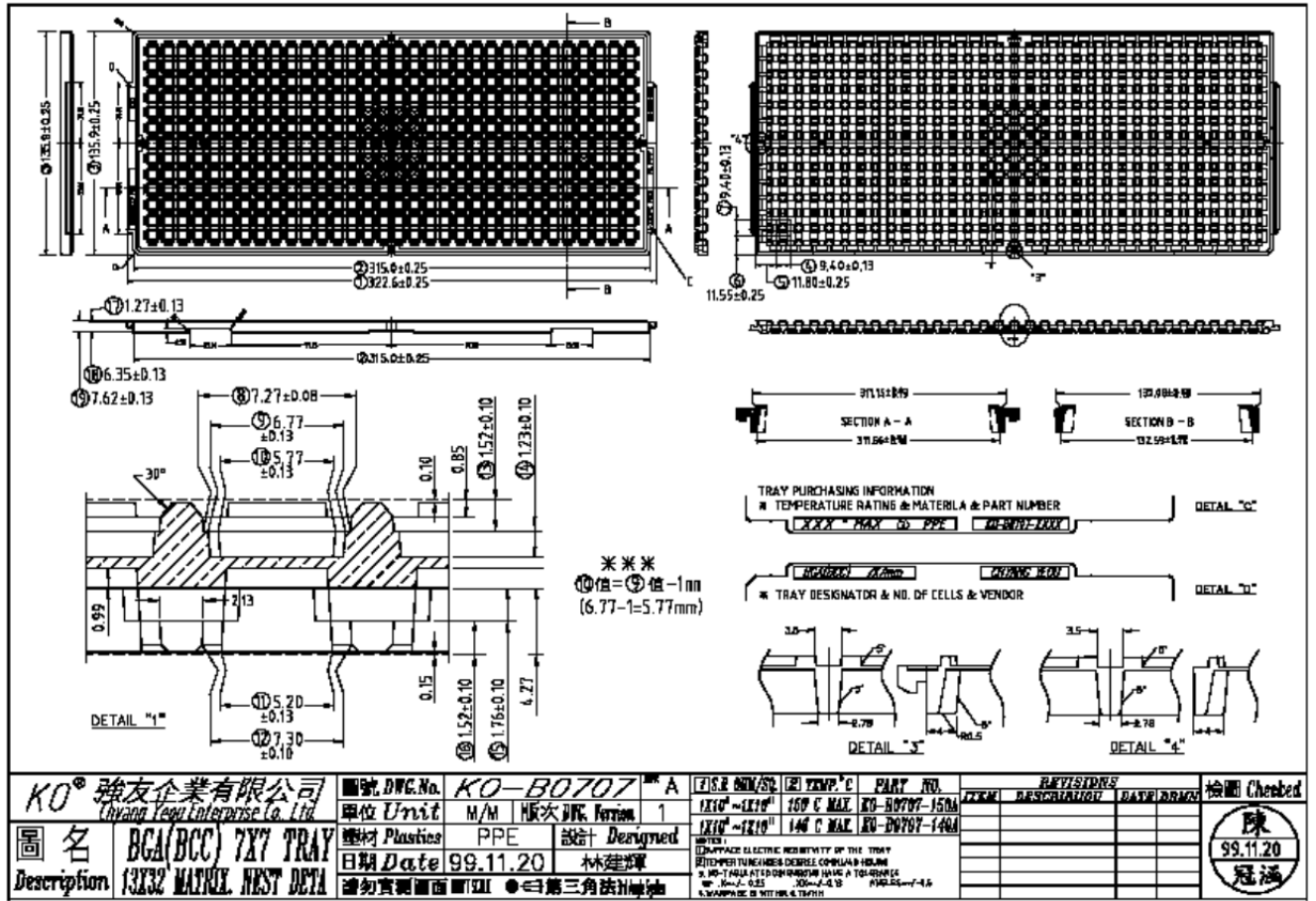


Figure 75: Tray C48/C44

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

Table 27: tray and tape & reel shipping products



6 Disclaimer

Important Notice The information in this document supersedes and replaces all previous versions of this document. The Information contained in this document is provided solely in connection with PIXCIR products. PIXCIR reserves the right to change the circuitry and specifications without notice at any time Purchasers are solely responsible for the choice, selection and use of the PIXCIR products and services described herein. PIXCIR assumes no liability whatsoever relating to the choice, selection or use of the PIXCIR and services described herein. No license, express or implied, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by PIXCIR for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein. UNLESS OTHERWISE SET FORTH IN PIXCIR TERMS AND CONDITIONS OF SALE, PIXCIR DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ITS PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED PIXCIR REPRESENTATIVE, PIXCIR PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. PIXCIR PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER S OWN RISK. Resale of PIXCIR products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by PIXCIR for the PIXCIR product or service described herein and shall not create or extend in any manner whatsoever, any liability of PIXCIR.