MPTT Architecture (v25) Update

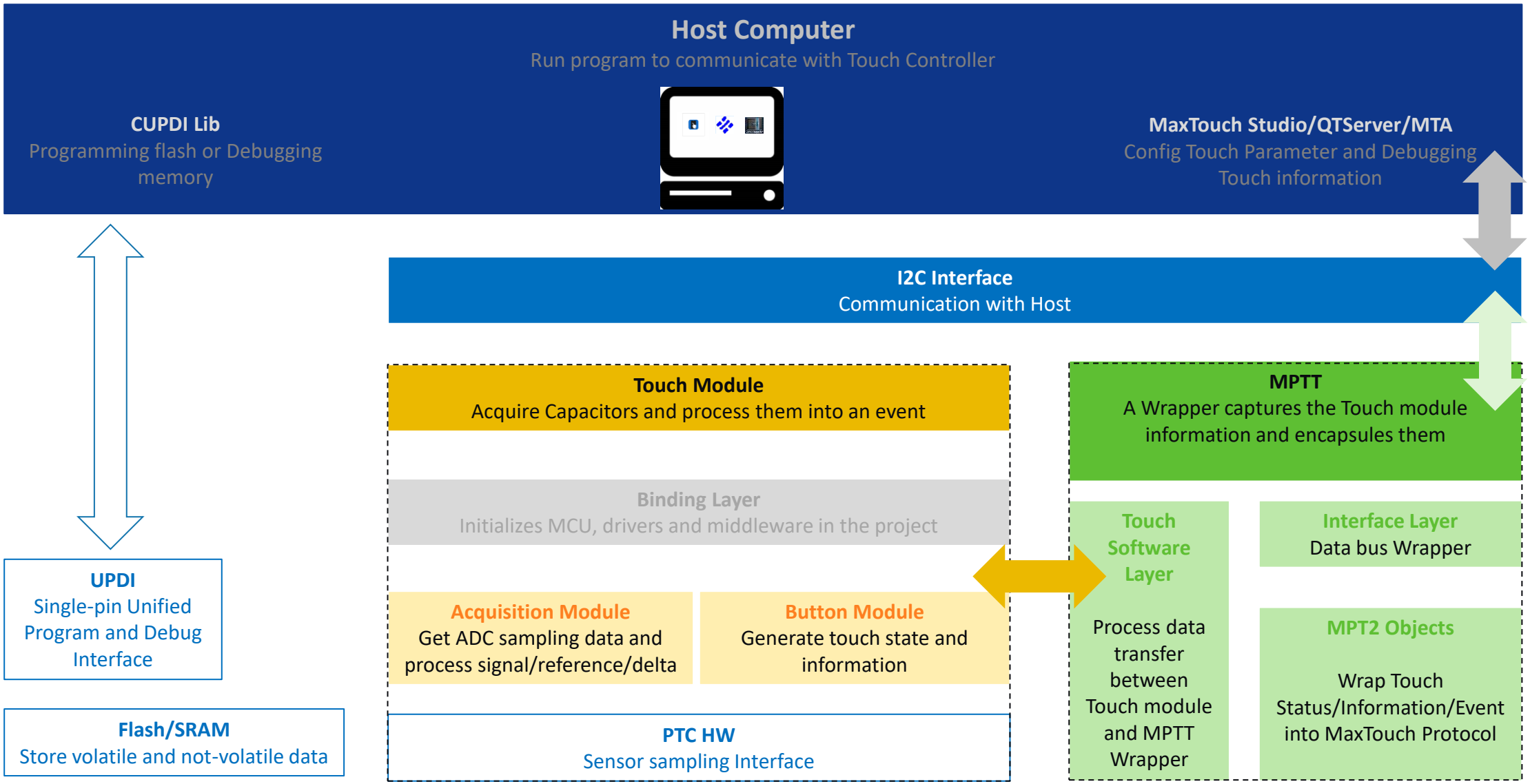
South China

Pitter Liao

2021 July

MICROCHIP

MPTT Architecture



MPTT Test Diagram

I2C Bridge

MTA

Walk all the Test items and log the Touch data for analysis

- FW Check and Upgrade
- Config Check and Download
- Silicon Version Check
- Self Test With each sensor
- UUID/Reference/Capacitance Logs

MPTT
Capture the Touch module state/data/information

Interface Layer
Data bus Wrapper

Touch Software Layer

Process data transfer between Touch module and MPTT Wrapper

MPT2 Object Table

Wrap Touch Status/Information into MaxTouch Protocol

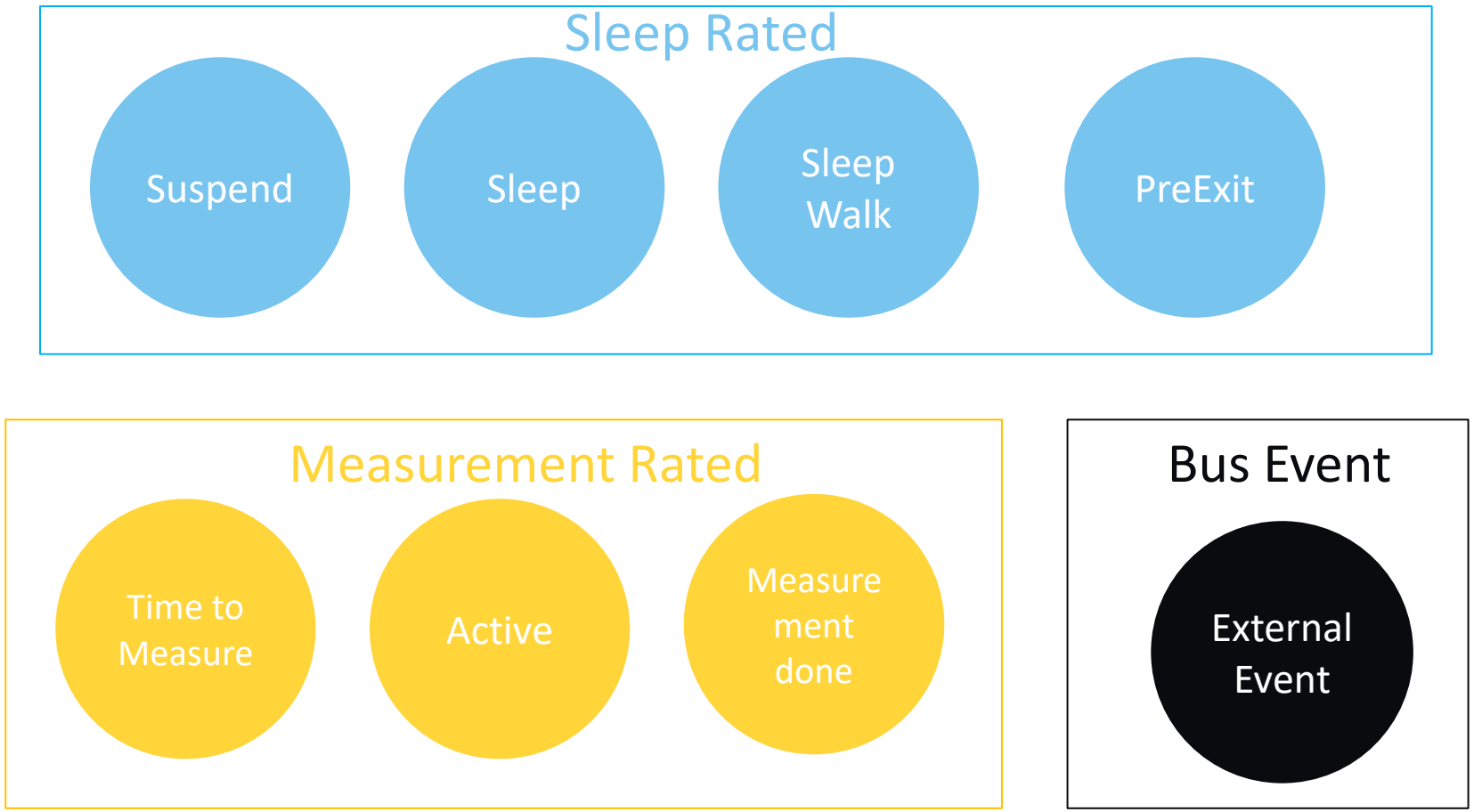
- **T37** Expose the Signal/Delta/reference/Cap
- **T6** Control for T37 output
- **T25** Process Selftest
- **T15** Button Process
- **T126** Low power consumption

- **Event system Sleep mode**
- Wakeup channel and threshold configurable
- Wakeup Key message filter
- Wakeup diagnostic data support

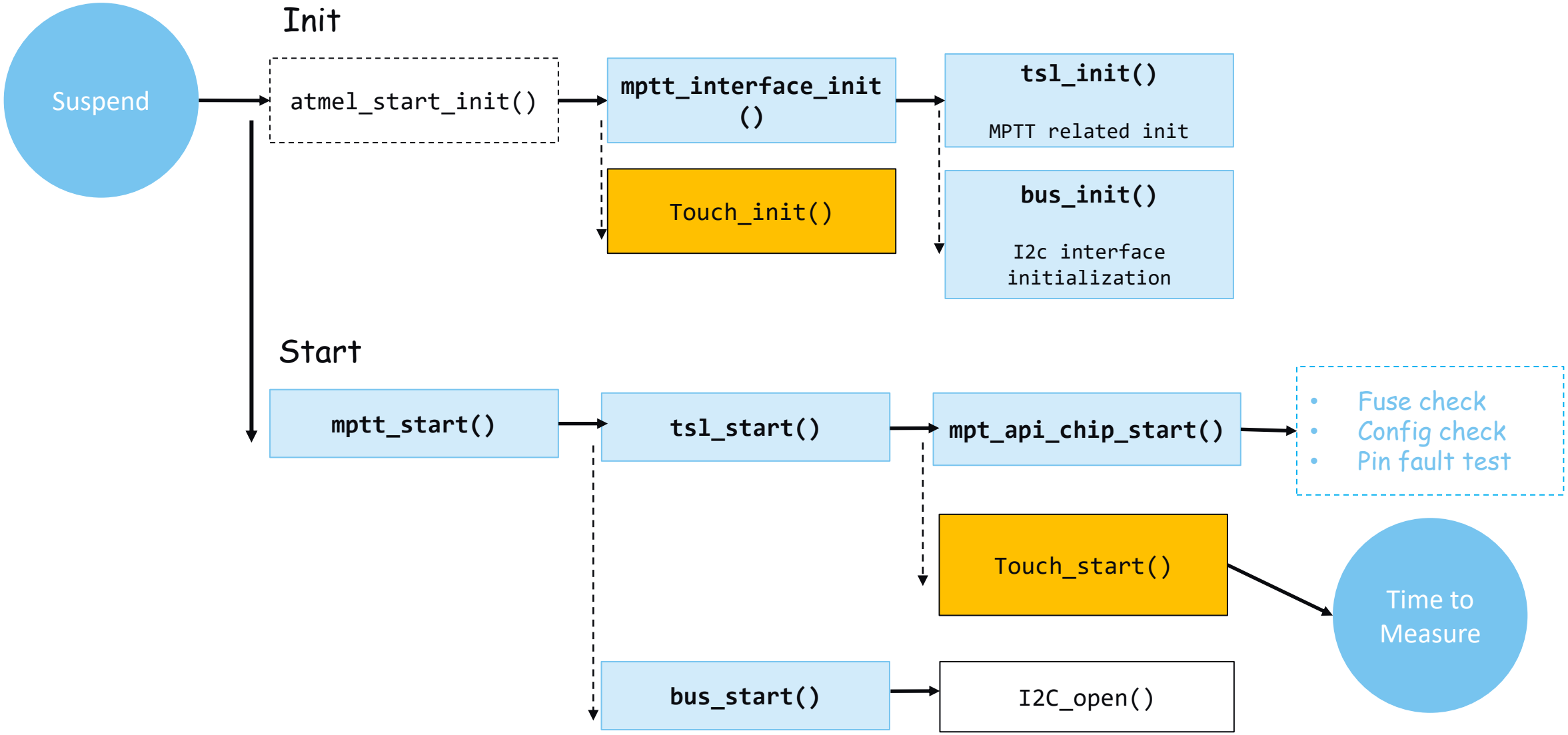
- **Software Sleep mode**
- Multi wakeup sensor channels appointment
- Wakeup key message filter
- Wakeup debug message and diagnostic support

MICROCHIP

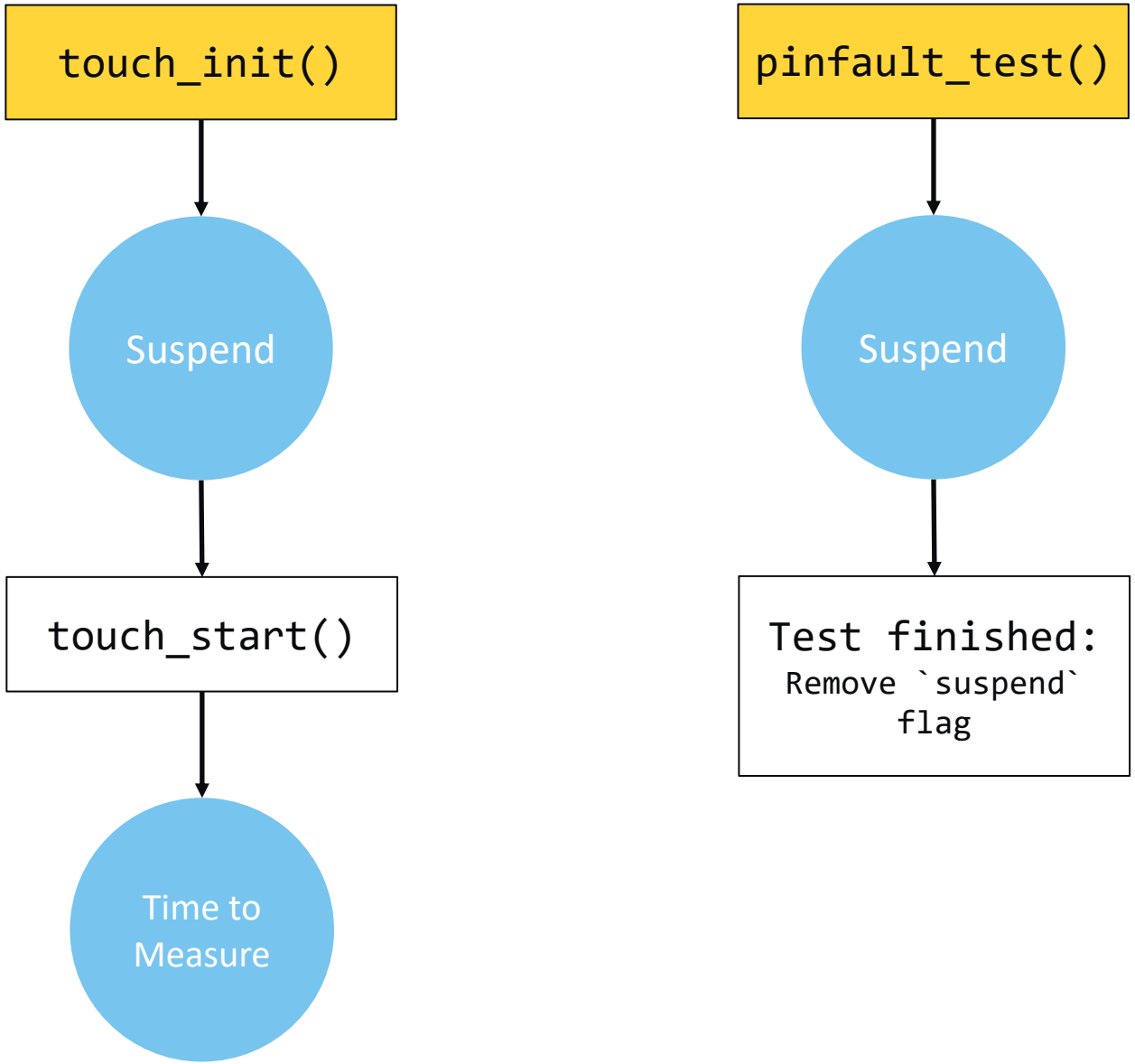
MPTT touch state machine (8 states)



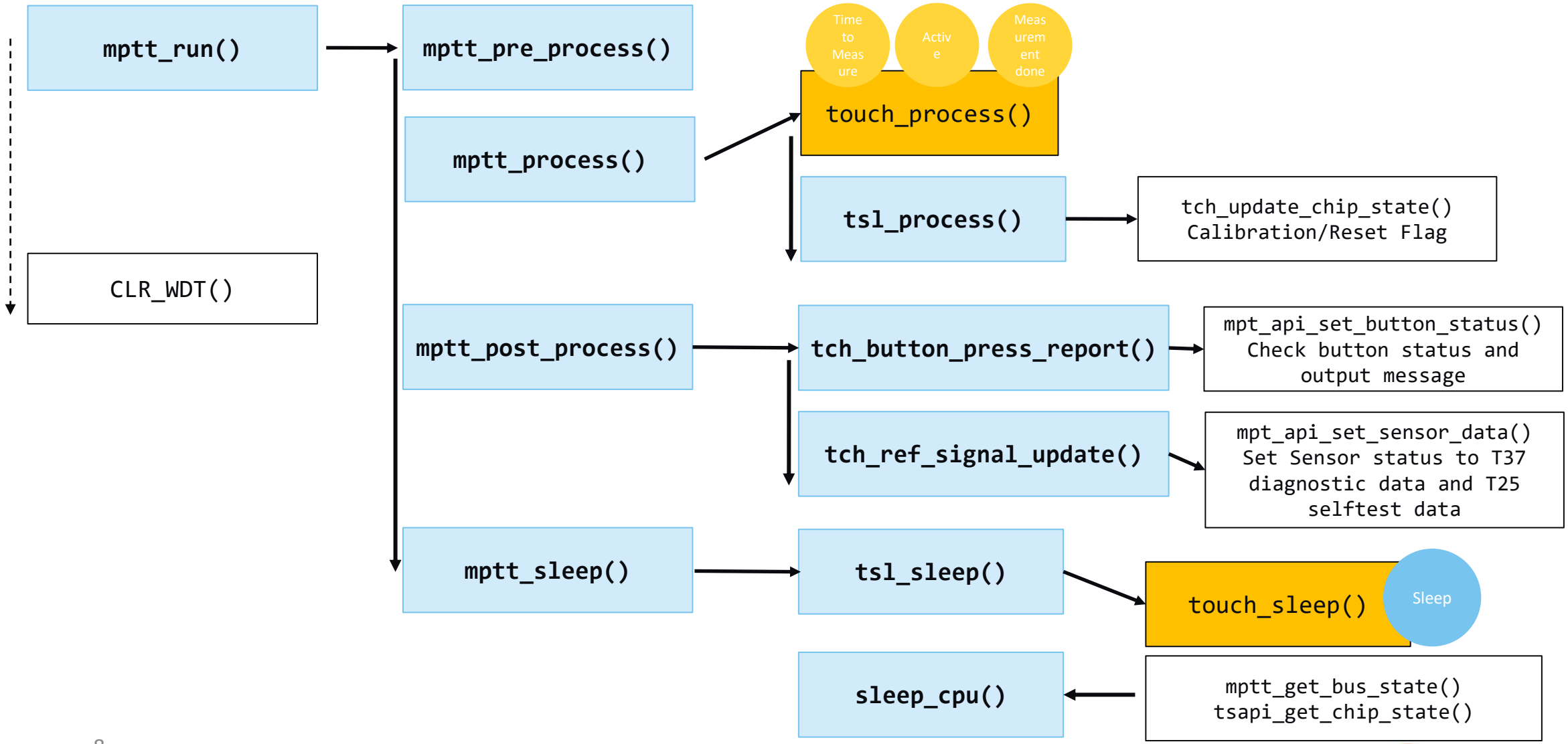
Bootup state machine



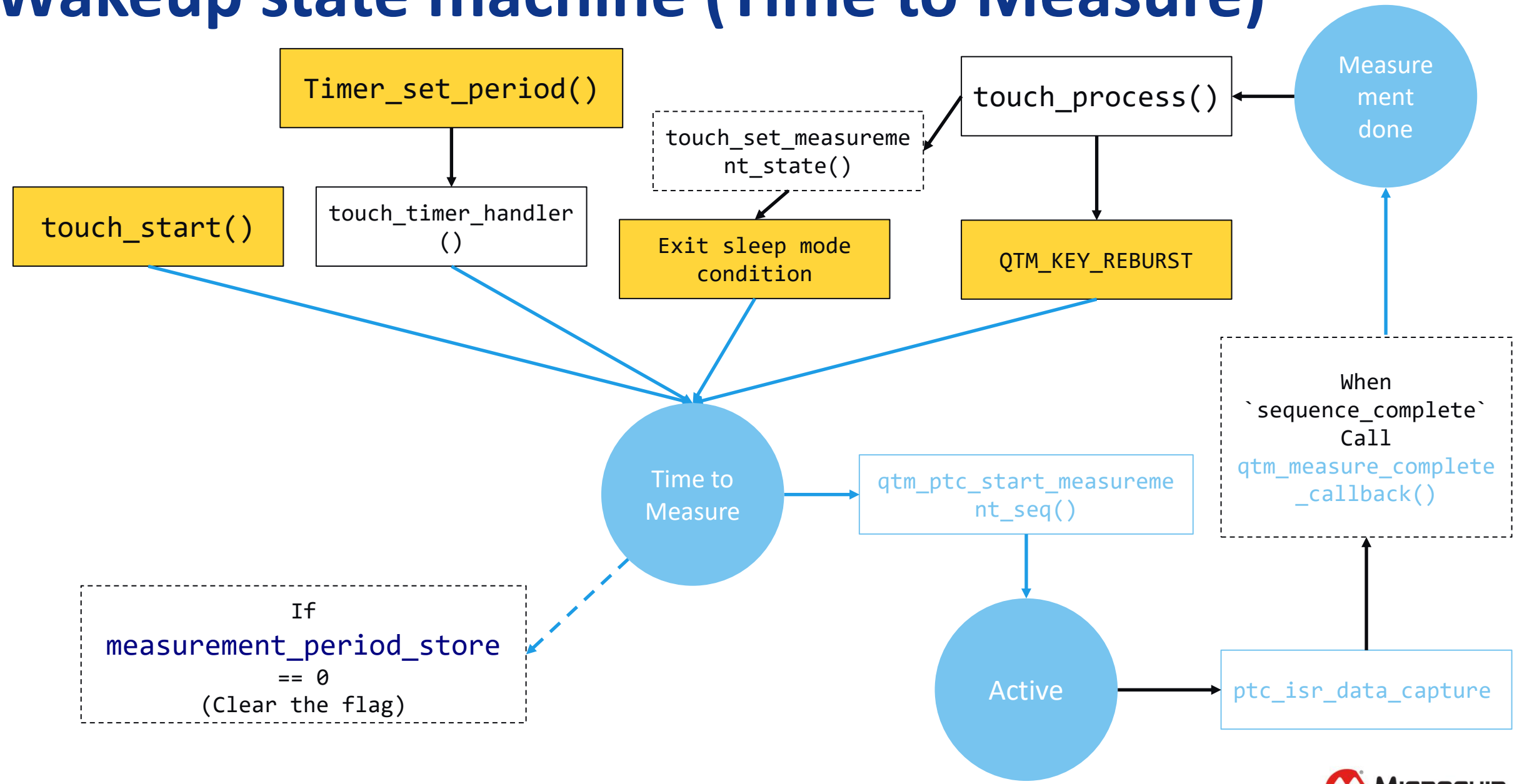
Wakeup state machine (Suspend)



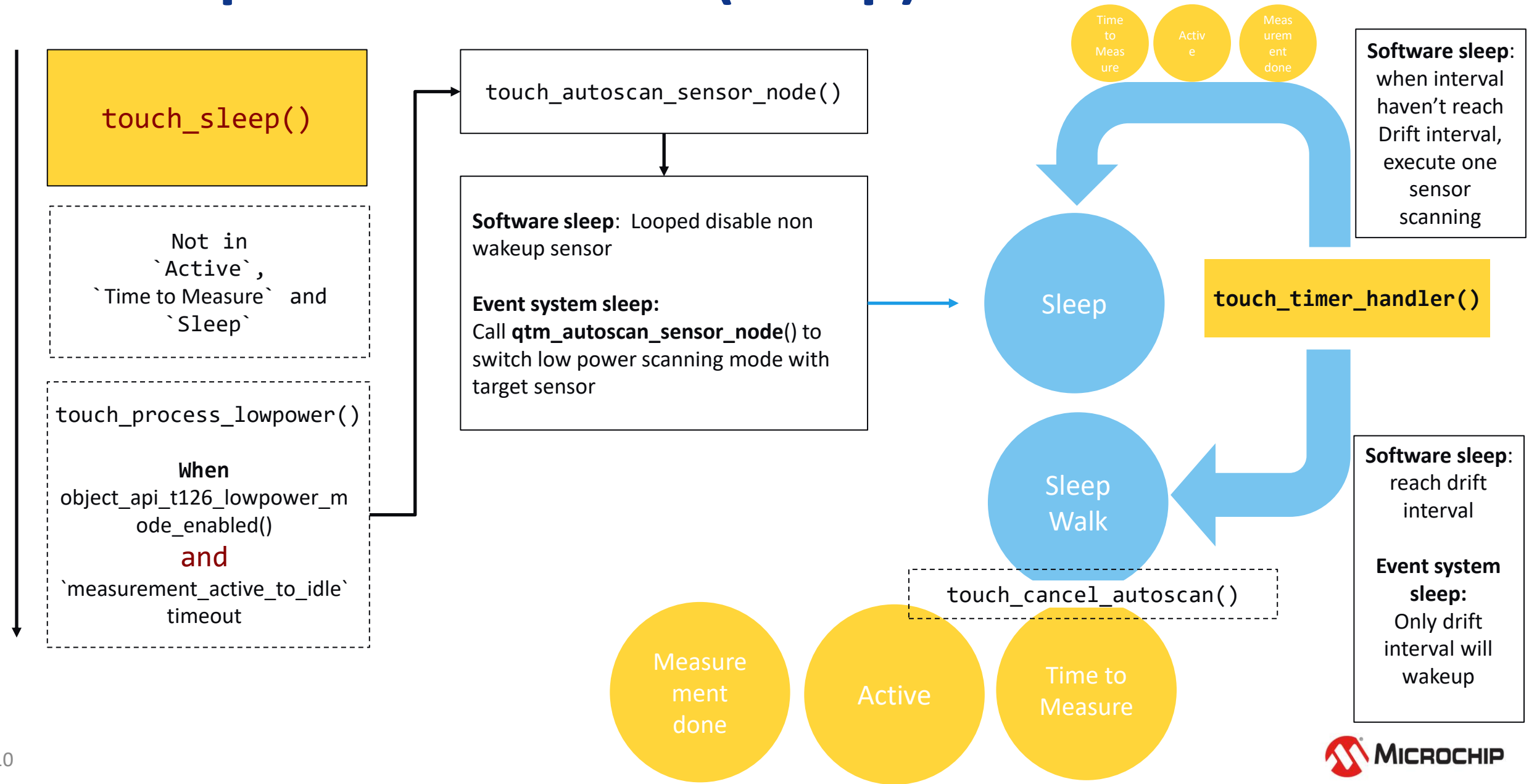
Main Loop state machine



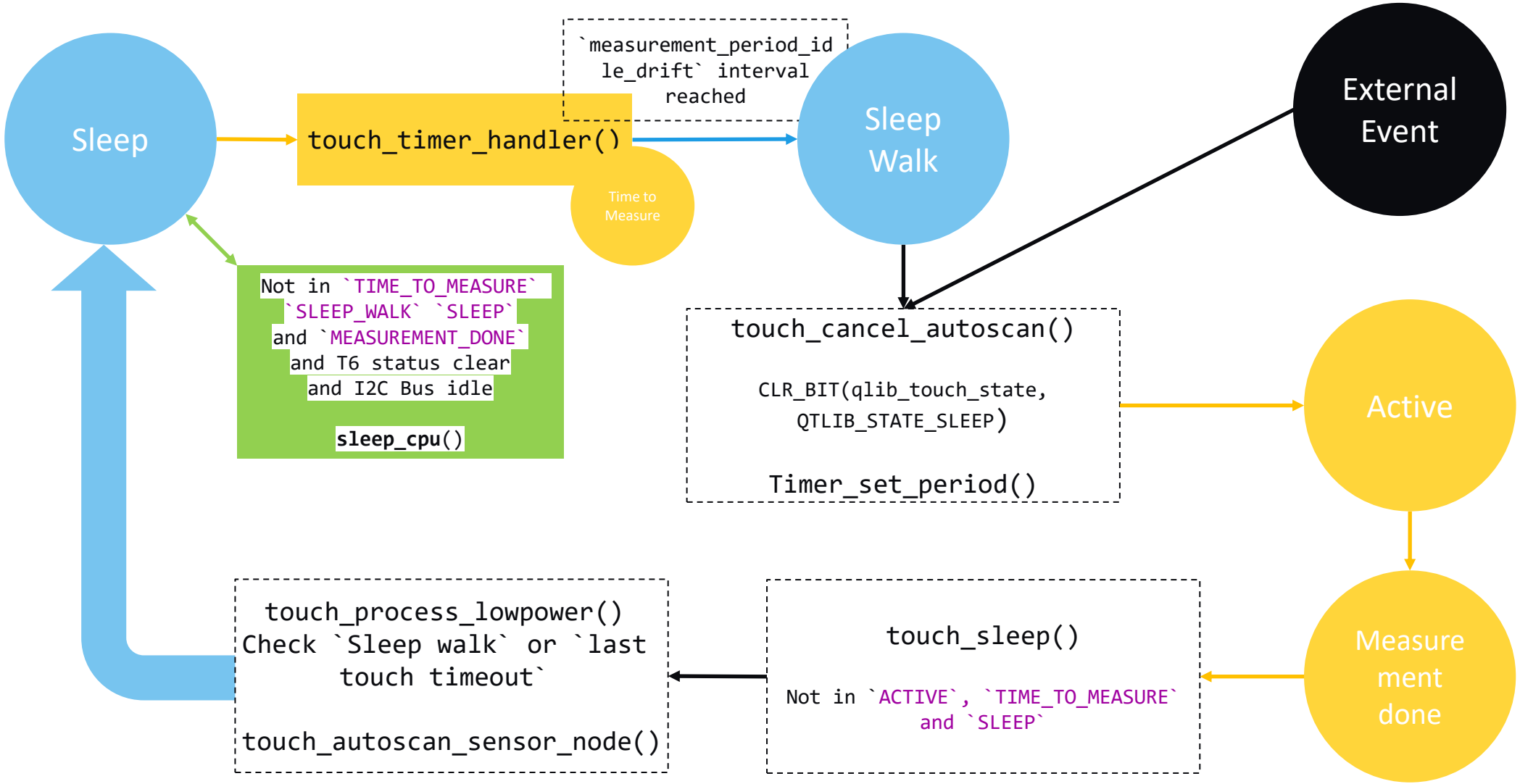
Wakeup state machine (Time to Measure)



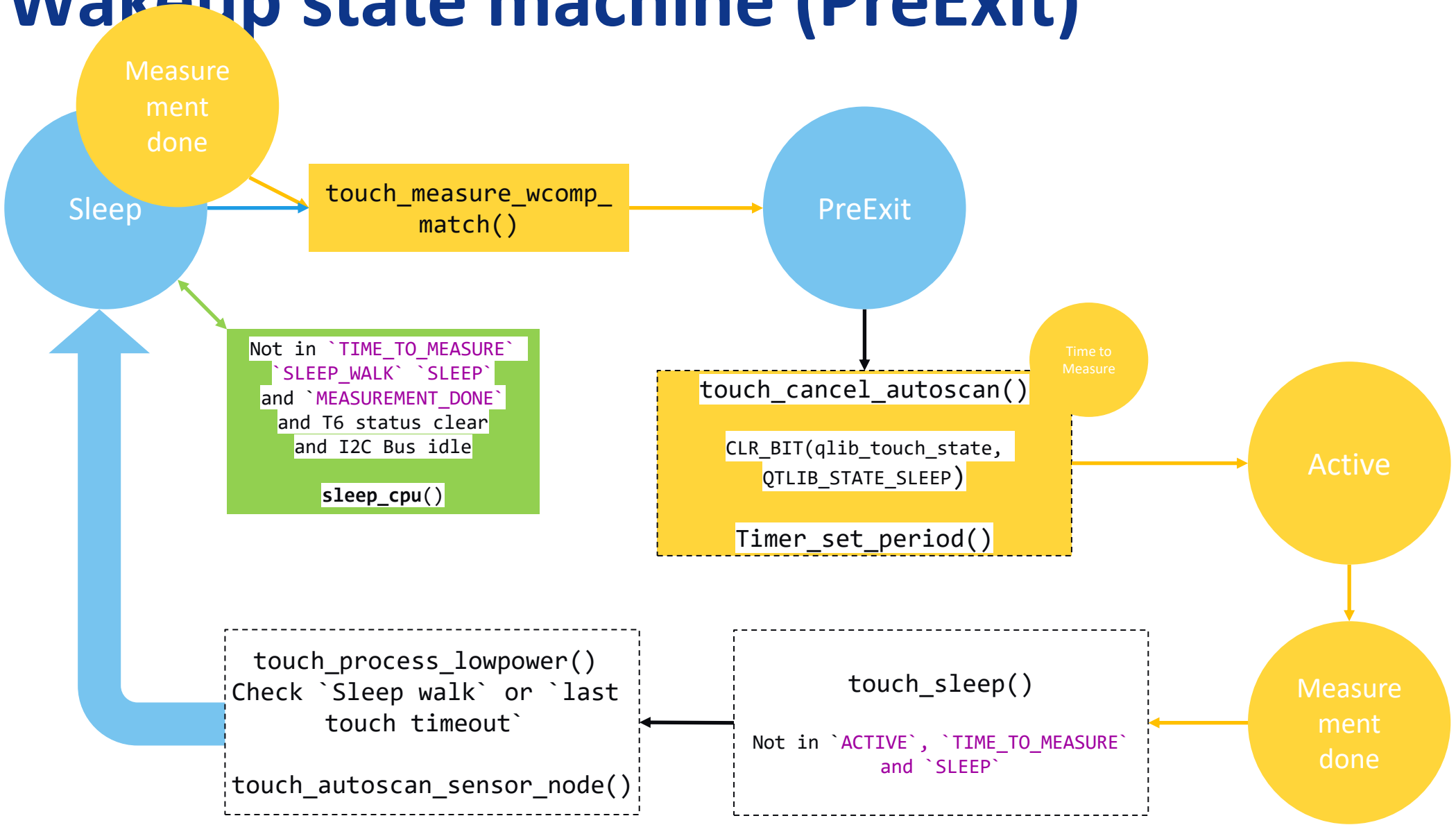
Wakeup state machine (Sleep)



Wakeup state machine (Sleep Walk)



Wakeup state machine (PreExit)



Power consumption

ATTINY3217 XPRO

3.3V power supply, CPU 8Mhz, I2C interface, BOD 1Khz, WDG Enabled 4s, Oversampling 16, Prescale div 2, CSD=2
(2 buttons, Self cap without DS, Freq Hop enabled, 2 buttons wakeup)

Software Sleep

	Interval (ms)	Active (uA)	Idle (uA)						1 button	4buttons	BOD sampled, ADC 8	BOD off	BOD on
Drift(s)		-	0.2	1	2	4	8	disabled	4	4	4	4	4
	1	2056	998	997	997	996	996	992					
	4	604	301	297	297	296	296	295					
	10	253	134	126	126	125	125	124					
	16	161	88	81	81	80	80	79					
	32	83	50	44	43	42	42	42					
	64	43	30	25	24	23	23	23	19	30	20	22	46.5
	128	23	20	15	14.3	13.8	13.7	13.3					
	255	13.7	16	10.5	9.5	9	8.8	8.5					
	disabled	3.3	3.5	3.4	3.4	3.3	3.3	3.2					

Event system Sleep

	Interval (ms)	Active (uA)	Idle (uA)						1 button	4buttons	BOD sampled, ADC 8	BOD off	BOD on
Drift(s)		-	0.2	1	2	4	8*	disabled	4	4	4	4	4
	1	2029	92	79	76	76	75	-					
	4	598	55	42	41	40	39	-					
	10	250	-	-	-	-	-	-					
	16	159	33	16	14.4	13.9	13.5	-					
	32	82	24	11.5	10.4	9.3	8.9	-					
	64	42	22	9.5	7.6	7	6.4	-	5.9	7.5	6.1	5.5	29
	128	23	22	8.3	6.6	5.8	5.4	-					
	255*	13.6	22	7.7	6.1	5.3	4.9	-					
	disabled	3.3	19.8	6.8	5.3	4.7	4.1	-					

Power consumption

ATTINY3217 XPRO

3.3V power supply, CPU 8Mhz, I2C interface, BOD 1Khz, WDG Enabled 4s, Oversampling 16, Prescale div 2, CSD=2
(1 buttons, Self cap without DS, Freq Hop enabled, 1 buttons wakeup)

	Software	Event System	Comments
Typical consumption (1 button@64ms)	20uA	7uA	Event system wakeup gets less consumption
Keys count limit	Get obvious power consumption as key count increased	Few power consumption increased	Software wakeup scanning execute all the code loop, more keys will cost more code execution time; Event system wakeup only execute the full code at drift cycle.
Wakeup keys limit	any number of wakeup keys could be set	Only 1 wakeup button could be set	Software wakeup loop scanning each each wakup keys. Event only support one target channel, If Event system want to wakeup by more key, need lump them with extra channel(But need consider the capacitor saturated)
Wakeup key scanning rate	Idle scanning rate * number of wakeup keys.	Idle scanning rate could be only set to (2^n) from 1 to 256ms	Software will get much slow scanning rate if more wakeup keys set because it's interlaced scanning.
Drift	Could disable	Must enable	

5 minutes to create Project

- Clone project from Github:

git clone https://github.com/PitterL/mpt2.git
git branch
git checkout EVK_3217_Xpro

ATtiny3217-1Finger-Low_Power-Project-MPT2_3217Xpro.zip

Source code address:

https://github.com/PitterL/mpt2/tree/EVK_3217_Xpro

- Files modified when transplanting to your own project:

qtouch\touch.h --- Define Qt library sensor configuration
mpt2\board.c --- Define MPTT firmware sensor layout

Steps 1 (qtouch\touch.h)

- Sensor channels:

```
#define DEF_NUM_CHANNELS 2
```

- Sensor configuration

```
/* Defines node parameter setting self cap
 * {Shield line, Y-line, Charge Share Delay, NODE_RSEL_PRSC(series resistor, prescaler), NODE_G(Analog Gain , Digital
 * Gain), filter level}
 */
#define NODE_0_PARAMS                                                                    \
    {                                                                                    \
        X_NONE, Y(3), 2, PRSC_DIV_SEL_2,                                                \
            NODE_GAIN(GAIN_1, GAIN_1), FILTER_LEVEL_16                                   \
    }
```

- Sensor selected

```
#define PTC_SEQ_NODE_CFG1   {   \
    NODE_0_PARAMS, NODE_1_PARAMS    \
}
```

- Keys Sensor configuration(Never mind, will be overridden in actual config)

```
/* Defines Key Sensor setting
 * {Sensor Threshold, Sensor Hysterisis, Sensor AKS}
 */
#define KEY_0_PARAMS                                                                     \
{                                                                                       \
    100, HYST_25, NO_AKS_GROUP                                                           \
}
```

- Keys selected

```
#define QTLIB_KEY_CONFIGS_SET { \
    KEY_0_PARAMS, KEY_1_PARAMS  \
}
```

Step 2 (mpt2\board.c)

- Definition button groups:

```
•   qbutton_config_t buttons_config[MXT_TOUCH_KEYARRAY_T15_INST] = {
•       #ifdef EVK_3217_XPLAIN
•       { .node = { .origin = 0, .size = 2 } },
•       #endif
•   };
// There are 2 groups default, you could set all channels to same group, or split it into several groups.
// It will finally be shown in `QTServer` T15 objects.
```

- Definition chip sensor channel group:

```
•   qtouch_config_t tsl_qtouch_def = {
•       #ifdef EVK_3217_XPLAIN
•       .matrix_nodes = {{.origin = 0, .size = 2}, {.origin =  2, .size = 2}},
•       #endif
•   };
// Sensor channels group as `X channels` * `Y channels` matrix format, it will show how many sensor matrix nodes in chip information.
// This is a virtual information, you just confirm xsize*ysize larger than qlib channels
```

Note:

origin: the start channel in qlib definition(touch.h)

size: how many sensor channels used in this group.

Compile and pack Firmware image

- Compiling with Microchip studio 7.0

https://www.microchip.com/en-us/development-tools-tools-and-software/microchip-studio-for-avr-and-sam-devices#tabs

with `Debug` options, there will be hex image at `Debug\ATtiny3217-1Finger-Low_Power-Project-MPT2.hex`

(You could use the image now)

For version management and combined image generated:

- Firmware build version and fuse definition:

qtouch\pack.h

```
The below definition is used for cupdi package tool.
    Firmware version is a 32bit hexi value, which will be packed into eeprom segment.
    Fuse content is a hexi byte array, NULL indicate ignored byte. THe information will be packed into fuse segment.
    Warning: Support '//' comment mark, but not support '/ * * /' comment mark inside the definition
*/
/* Project code*/
#define PROJECT_CODE 0x5630323512   /*V025, v1.2*/

/* Fuse content */
// BOD level 2(2.6v Sampled 1Khz at Sleep, Enabled at Active), OSC 16Mhz, NVM protect after POR, EEPROM erased, WDT(4096ms)
#define FUSES_CONTENT {0x0A, 0x46, 0x7D, 0xFF, 0x00, 0xF6, 0xFF, 0x00, 0x00, 0xFF, 0xC5 }   /* BYTE order */
```

Compile and pack Firmware image

- Pack with CUPDI tool:

https://github.com/PitterL/cupdi

You could get cupid of `window` or `linux` version, after that use it to pack the hex file with fuse information

cupdi.exe -d tiny3217 -f “<Path:\\>ATtiny3217-1Finger-Low_Power-Project-MPT2.hex” --pack-build

Then , The packed firmware(.ihex) file will come out:

```
D:\Users\xxx\vs\cupdi\cupdi_win_v1.16b\x64>cupdi -d tiny3217 -f "D:\Users\xxx\atmel studio\ATtiny3217-1Finger-Low_Power-Project-MPT2\ATtiny3217-1Finger-Low_Power-Project-MPT2\Debug\ATtiny3217-1Finger-Low_Power-Project-MPT2.hex" --pack-build
Fuse[0]: 0a
Fuse[1]: 46
Fuse[2]: 7d
Fuse[3]: ff
Fuse[4]: 00
Fuse[5]: f6
Fuse[6]: ff
Fuse[7]: 00
Fuse[8]: 00
Fuse[9]: ff
Fuse[10]: c5
Saved Hex to "D:\Users\xxx\atmel studio\ATtiny3217-1Finger-Low_Power-Project-MPT2\ATtiny3217-1Finger-Low_Power-Project-MPT2\Debug\ATtiny3217-1Finger-Low_Power-Project-MPT2.ihex"
```

Flashed the image

- **For .hex file, you could use ICE or any other debug tool for download image.**
- **Now I teach you how to flash combined image(.ihex) and tuning**

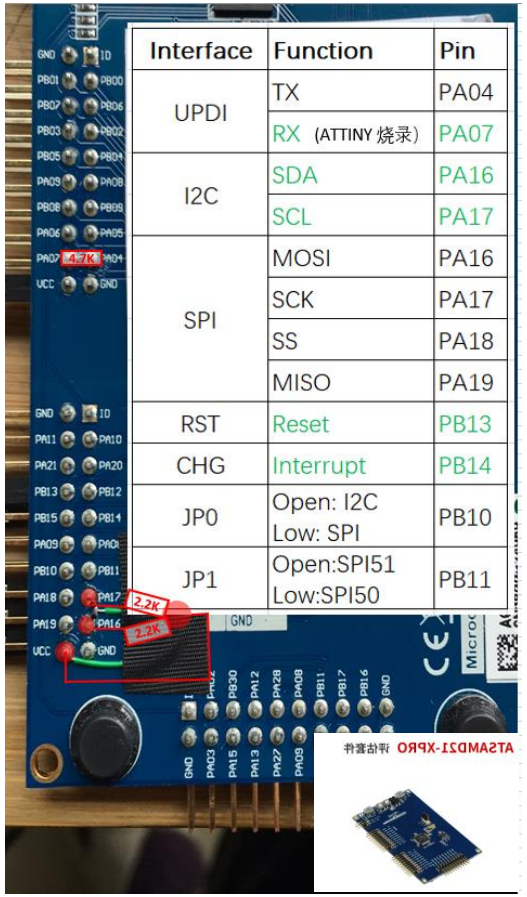
Prepare SAMD21 Xpro board for a tuning bridge:

https://www.microchipdirect.com/dev-tools/ATSAMD21-XPRO?allDevTools=true

Modify the hardware with 3 resistors, and download the image:

D21UsbBridgeAsf_20210612.hex

Interface	Function	Pin	硬件改动	说明
UPDI	TX	PA04		悬空
	RX	PA07	串一个4.7K电阻到TX	接ATTiny芯片的Reset/UPDI脚，烧录固件用
I2C	SDA	PA16	并一个2.2K电阻到VCC	I2C通信接口
	SCL	PA17	并一个2.2K电阻到VCC	
SPI	MOSI	PA16		SPI通信接口
	SCK	PA17		
	SS	PA18		
	MISO	PA19		
RST	Reset	PB13		使用ATTiny芯片时候不连接，使用MaxTouch时候接Reset脚
CHG	Interrupt	PB14		接触摸芯片中断脚
VCC	Power	-		给触摸芯片提供3.3v supplier
GND	GND	-		和触摸芯片共地Gound
跳线0	选择接口模式	PB10		悬空:I2C模式，接地:SPI模式
跳线1	SPI通信协议	PB11		悬空:SPI51，接地:SPI50



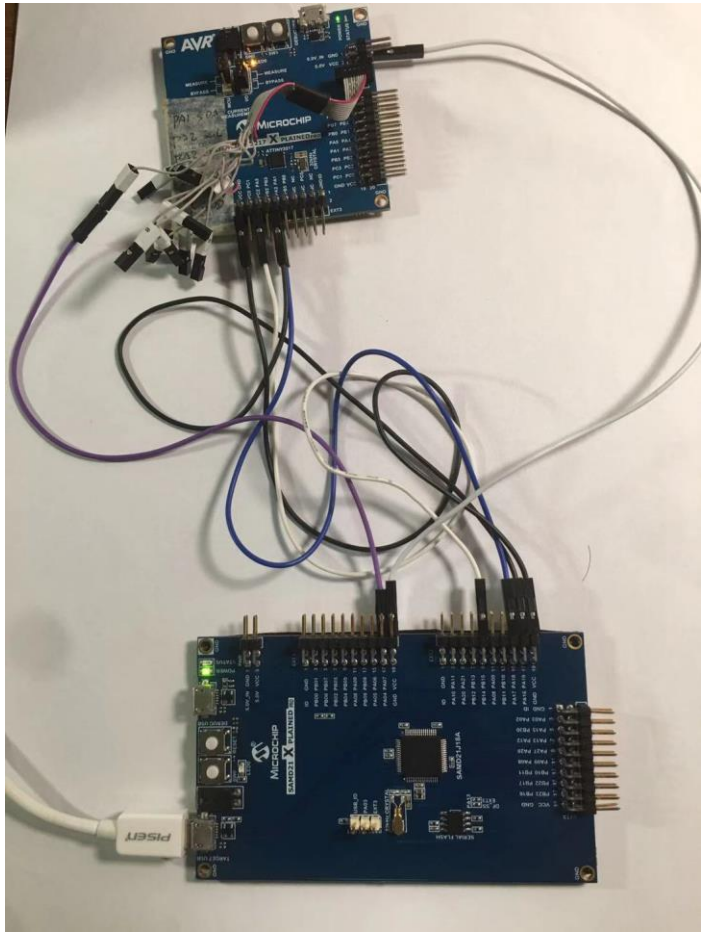
Interface	Function	Pin
UPDI	TX	PA04
	RX (ATTINY 烧录)	PA07
I2C	SDA	PA16
	SCL	PA17
SPI	MOSI	PA16
	SCK	PA17
	SS	PA18
	MISO	PA19
RST	Reset	PB13
CHG	Interrupt	PB14
JP0	Open: I2C Low: SPI	PB10
JP1	Open:SPI51 Low:SPI50	PB11

MICROCHIP

Flashed the image

- **After you create a tuning bridge, you could connect the 6 wires as below:**

Net	Name	D21 Xpro	3217Xpro
UPDI	RX	PA07	UPDI
I2C	SDA	PA16	PA01
	SCL	PA17	PA02
CHG	INT	PB14	PA03
VCC	Power	VCC	VCC
GND	GND	GND	GND



Now you could use the CUPDI flash command:

cupdi.exe -d tiny3217 -c com14 -f “<Path:\\>ATtiny3217-1Finger-Low_Power-Project-MPT2.ihex” --program -v 2

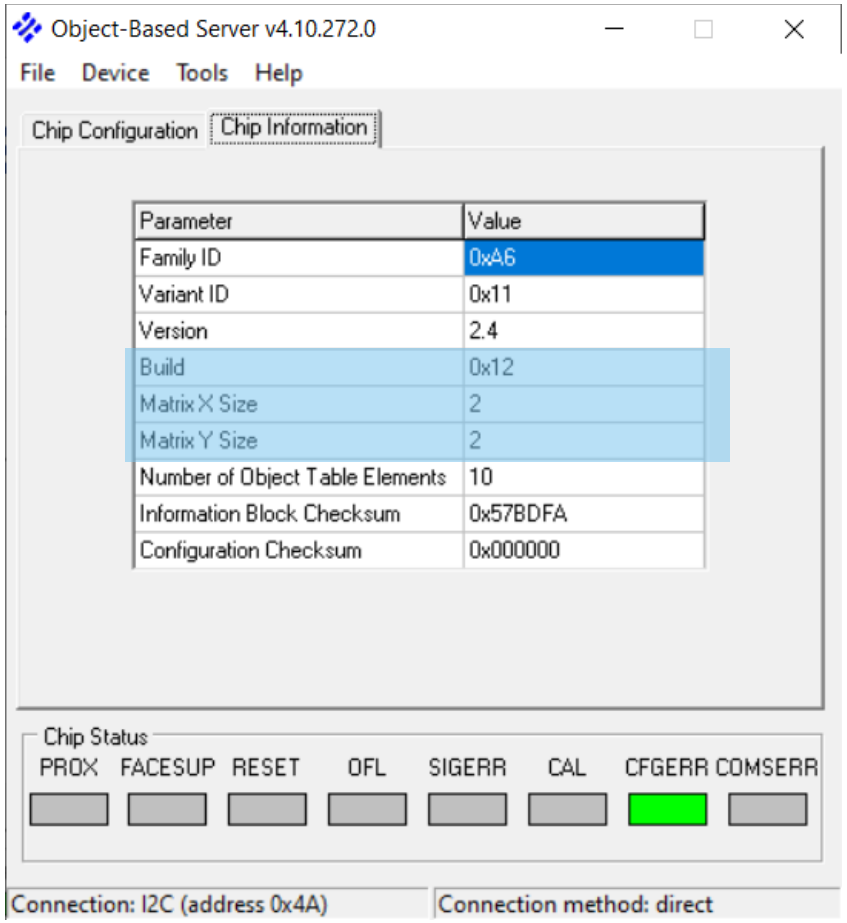
Flashed the image

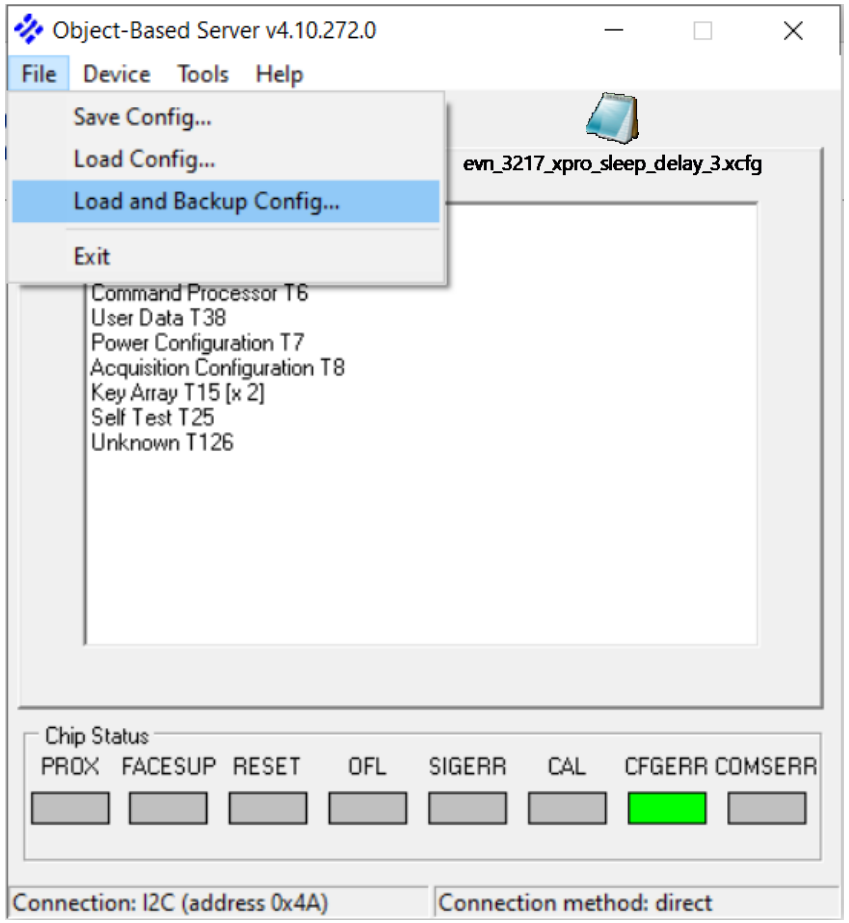
- **Logs:**

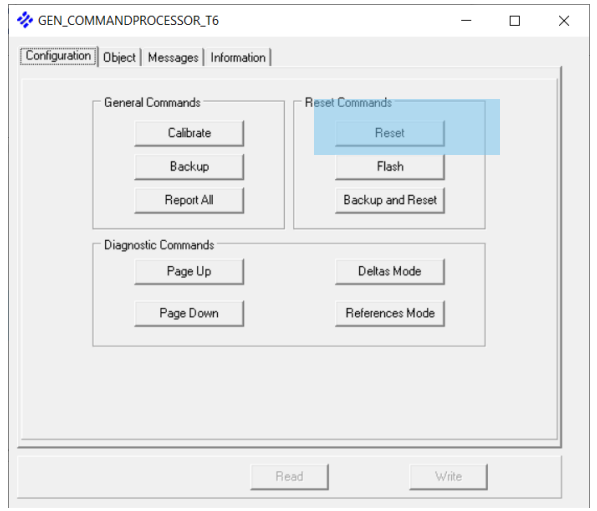
```
D:\Users\xxx\vs\cupdi\cupdi_win_v1.16b\x64>cupdi -d tiny3217 -f "D:\Users\xxx\atmel studio\ATtiny3217-1Finger-Low_Power-Project-MPT2\ATtiny3217-1Finger-Low_Power-Project-MPT2\Debug\ATtiny3217-1Finger-Low_Power-Project-MPT2.ihex" -c com14 --program -v 2
<NVM> init nvm
<NVM> Reading device info
<NVM> Entering NVM programming mode
<NVM> Erase device
<NVM> Write Auto
<NVM> Writes to flash
Writing flash page(0/172) at 0x8000
Writing flash page(1/172) at 0x8080
Writing flash page(2/172) at 0x8100
Writing flash page(3/172) at 0x8180
...
Writing flash page(168/172) at 0xd400
Writing flash page(169/172) at 0xd480
Writing flash page(170/172) at 0xd500
Writing flash page(171/172) at 0xd580
<NVM> Write Auto
<NVM> Writes to eeprom
Writing eeprom page(0/1) at 0x1300
<NVM> Write Auto
<NVM> Writes to fuse(hex) [1280]: 0A
<NVM> Writes to fuse(hex) [1281]: 46
<NVM> Writes to fuse(hex) [1282]: 7D
<NVM> Writes to fuse(hex) [1283]: FF
<NVM> Writes to fuse(hex) [1284]: 00
<NVM> Writes to fuse(hex) [1285]: F6
<NVM> Writes to fuse(hex) [1286]: FF
<NVM> Writes to fuse(hex) [1287]: 00
<NVM> Writes to fuse(hex) [1288]: 00
<NVM> Writes to fuse(hex) [1289]: FF
<NVM> Writes to fuse(hex) [128A]: C5
Program finished
<NVM> Leaving NVM programming mode
<NVM> deinit nvm
```

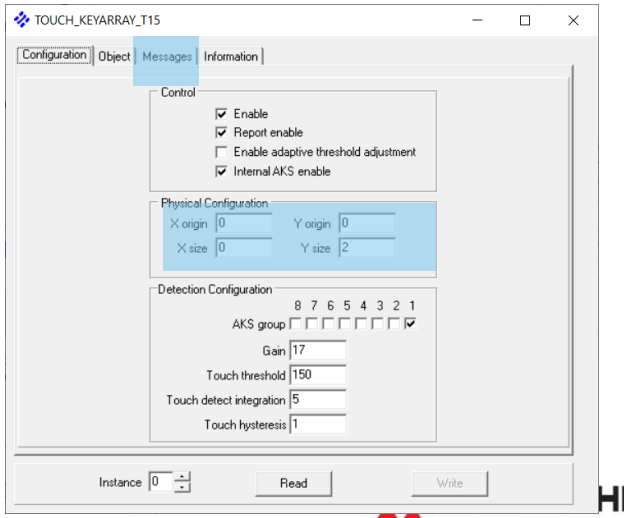
Use QTServer loading config

- **Use QTServer loading config**
- **Load and Backup config->select config file->(After loaded)->T6->Reset**









Save out production image

Now the data in 3217 will have full information：

Memory layout -

- 0x8000: Flash，Store program
- 0x1300: User，Store program extra information like FW crc
- 0x1400: EEPROM，Store config information
- 0x1280: Fuse

You could save out production image:

cupdi.exe -d tiny3217 -c com14 -f “<Path:\\>ATtiny3217-1Finger-Low_Power-Project-MPT2.save” --save

- It will save out the “.save” file with hex format, that could be use as offline flash chip.

Log:

D:\Users\xxx\vs\cupdi\cupdi_win_v1.16b\x64>cupdi -d tiny3217 -f "D:\Users\xxx\atmel studio\ATtiny3217-1Finger-Low_Power-Project-MPT2\ATtiny3217-1Finger-Low_Power-Project-MPT2\Debug\ATtiny3217-1Finger-Low_Power-Project-MPT2.ihex" -c com14 --save -v 2

<NVM> init nvm

<NVM> Reading device info

<NVM> Entering NVM programming mode

<NVM> Read memory 0x1300 size 4(0x4)

<NVM> Read memory 0x1300 size 20(0x14)

<NVM> Read memory 0x8000 size 22003(0x55f3)

<NVM> Read memory 0x1400 size 256(0x100)

<NVM> Read memory 0x1280 size 11(0xb)

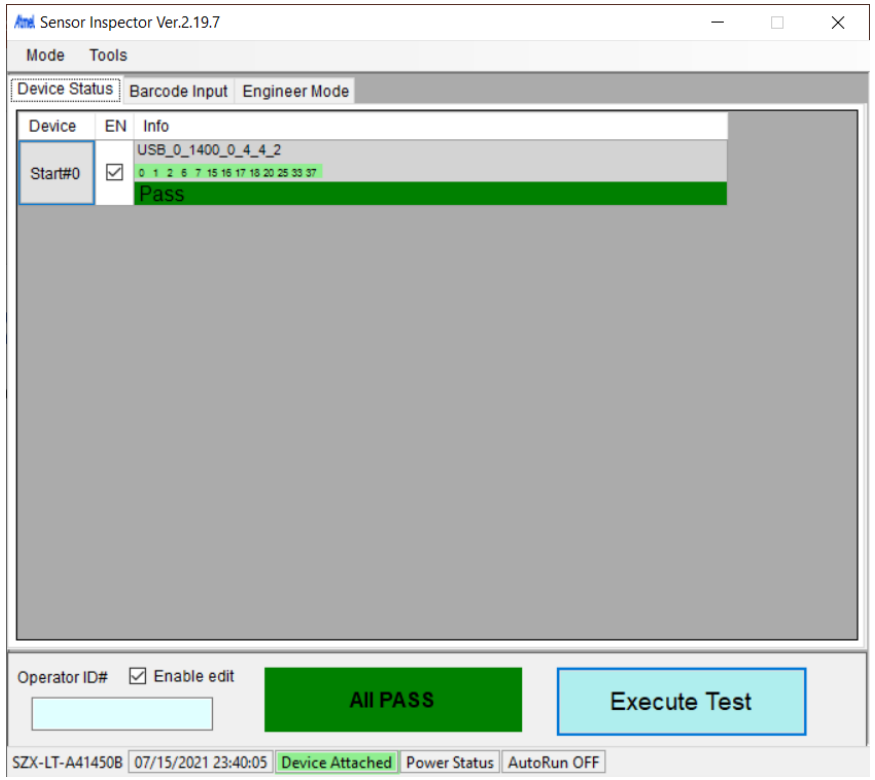
Save Hex to "D:\Users\xxx\atmel studio\ATtiny3217-1Finger-Low_Power-Project-MPT2\ATtiny3217-1Finger-Low_Power-Project-MPT2\Debug\ATtiny3217-1Finger-Low_Power-Project-MPT2.save"

<NVM> Leaving NVM programming mode

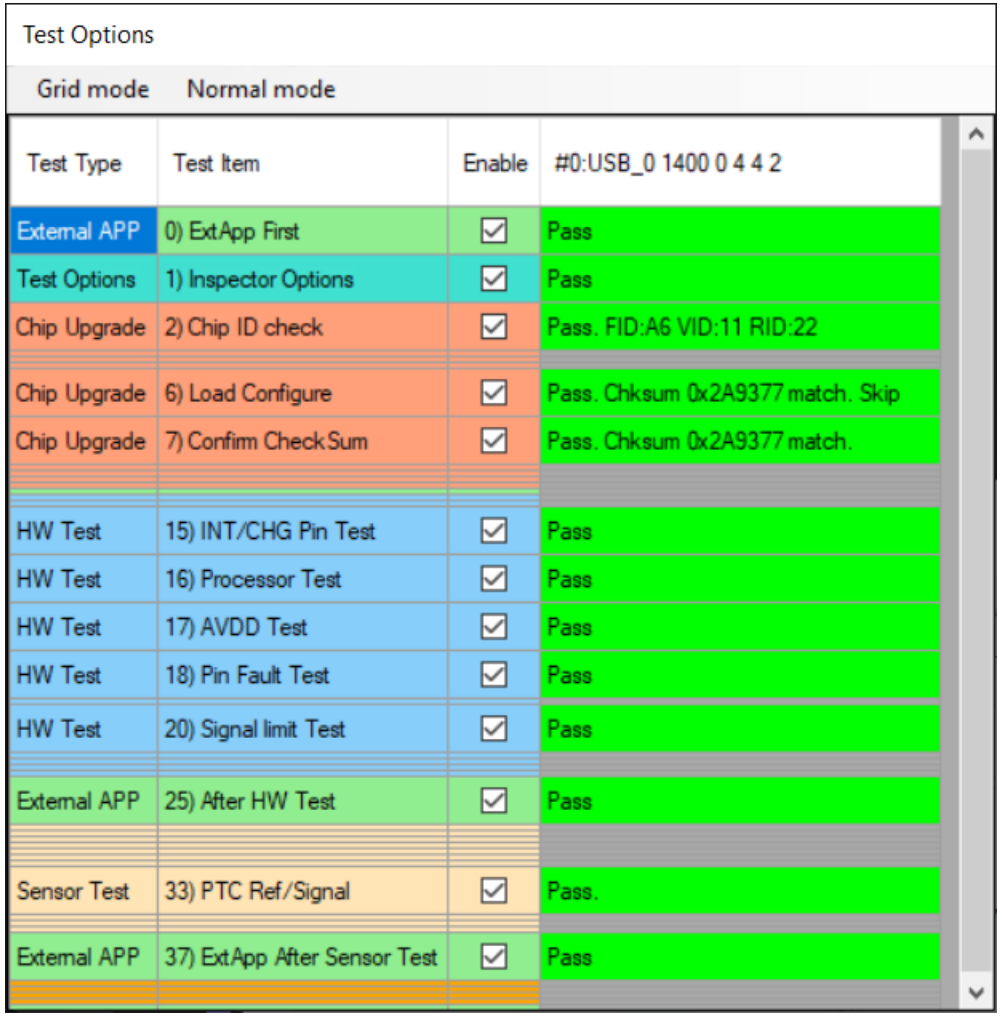
<NVM> deinit nvm

Use MTA to production line test

- **The MTA (v2.19.7-sp) production tool support:**
  - Firmware upgrade
  - Silicon ID check
  - Config upgrade
  - Pin test
  - Reference/Capacitance check



MaxTouch_Analyzer_V2.19.7_3217Xpro.7z



Sensor_SZX-LT-A41450B_0715_233914_514E59515220C4B12320_PASS.csv.txt

The MPTT (v25) Feature:

- **Sensor Pin auto config**
  - When clone the project, no need to set each pin config, firmware will auto config used pins.
- **Fuse verification**
  - Notice a message if fuse mis-match between code setting and current chip setting.
- **Signature Row output**
  - Output chip signature row that will record as chip UUID.
- **I2C bus monitor(Software type)**
  - Support I2C latched monitor
- **Watch Dog**
  - Support Watch Dog and will automatically calculate drift interval
- **Reference/Capacitance/Delta debug**
  - Out put RSD debug data at runtime environment through I2C
- **Low Power function(Event system and soft sleep)**
  - Support Low Power function dynamically enable with debug status
- **Self Test with Pin fault/capacitance/Reference**
  - Support self test command both normal and low power mode
- **Calibration**
  - Support calibration command both normal and low power mode

The MTA (v2.19.7-sp) Feature:

- **External command at first step for firmware download before test**
  - Could execute firmware download and production test one step
- **Chip UUID for Signature Row data log record**
  - Record the UUID in test log that could look back the test log.
- **MPTT compatible issue, there is no extra objects required at Test**
  - Support minimum objects required for firmware
- **PTC test items extension available for more keys support**
  - Configurable test button list in XML file
- **Test log more readable**
  - Test log information more clear

Thank you!