**Project Name: SMART I 6DI**

**Version: 1.02**

Document Originally Prepared by: Ubayathulla

Document Prepared on: 10/04/24

Document Last Modified on: 24/05/2024

This is a confidential document that should be circulated only within Elmeasure India Pvt Ltd, Bangalore. This document is of a proprietary nature and not licensed for resale or distribution. No part of this document shall be reproduced or used without prior written permission of the owner. Any commercial document if found similar to the above document with the Individual or the organization shall be punishable with severe civil and criminal penalties to the maximum extent by the International Law.

All Rights of these Documents are reserved with ElMeasure India Pvt Ltd.,

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Date | Reason for change | Change owner | Revision |
| 01 | 10/04/24 | Initial specification | Ubayathulla A | 00 |
| 02 | 15/05/24 | Power supply specification |  | 01 |
| 03 | 24/05/24 | Power supply schematic added |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Contents (Shall be updated Later)**

[1. Introduction: 4](#_heading=h.2et92p0)

[2. Block Diagram: 4](#_heading=h.tyjcwt)

[**4. Architecture Hardware 10**](#_heading=h.6weqwihfoo1b)

[5. Architecture Firmware 10](#_heading=h.1t3h5sf)

## Introduction:

The main purpose of this document is to explain the design requirement and Implementation of the SMART I – 6DI.

The SMART I – 6DI collects the number of pulses from different devices like water meters, Gas meters, etc., which have an option of pulse output. The same is communicated to the masters with UART and SPI interface based on the requirement. The UART has the option of RTS to communicate with RS485

## Block Diagram:

**Hardware**

1. 40V / 300V AC/DC input and 5V output with 150mA
2. 5V to 3.6V LDO with 150mA
3. MSPM0L1106 Microcontroller

**2.1 Block Diagram:**

40V-300V AC/DC to 5V

5V to 3.6V

**Microcontroller**

**MSPM0L1106**

UART

SPI

DI1

DI0

DI2

DI3

DI4

DI5

PWR

DI

Status

EXT Comm

Status

**2.1.1 Power Supply Requirements:**

* **Input Power + 240V AC/DC (External)**

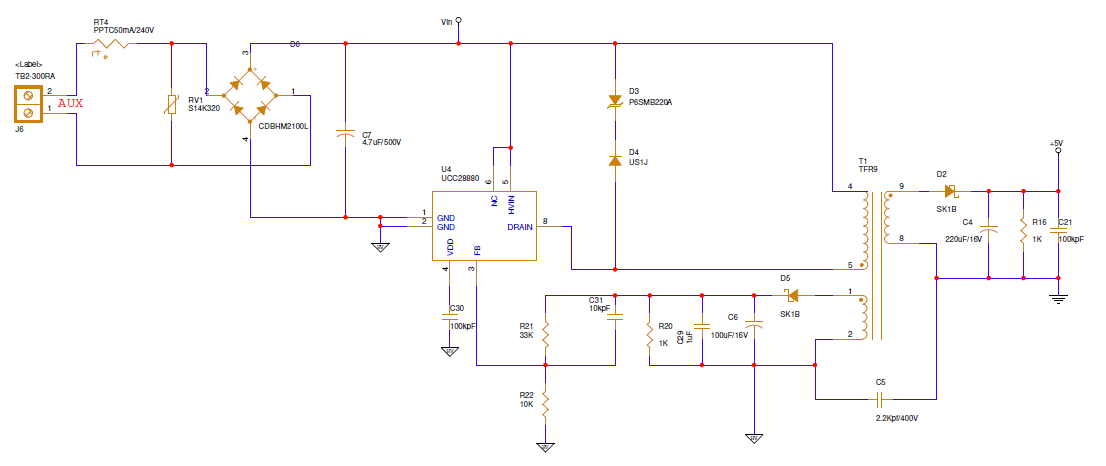
This external power source of **+ 240V** is applied to the SMART I – 6DI controller.

Acceptable Level:

**MIN Voltage: +40V AC/DC**

**MAX Voltage: + 300V AC/DC**

Measure across: J2 AUX terminal block



* **Input Power + 240V (External)**

This external power source of + **240V** is applied to the SMART I – 6DI controller.

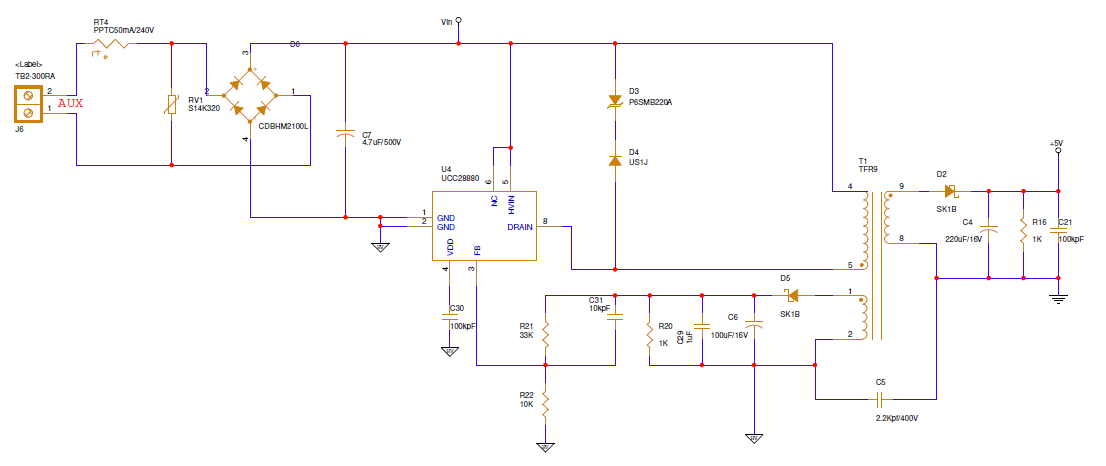
Acceptable Level:

MIN Voltage: + 40V AC/DC

MAX Voltage: + 300V AC/DC

Current Output = 150mA

Measure across < Shall be updated later>



* **Input Power + 5V DC (External)**

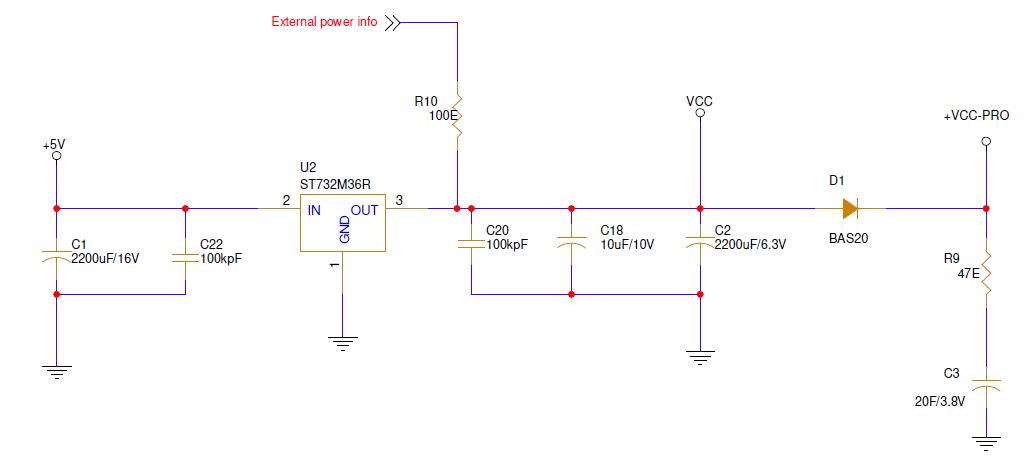
This external power source of + 5V is applied to the SMART I – 6DI controller.

Acceptable Level:

MIN Voltage: +3.8V DC

MAX Voltage: + 6.2V DC

Current Output = 150mA



* **Input Power + 3.6V DC (Internal)**

This internal power source of + 3.6V is applied to VDD pins of MSPM0L1107 Microcontroller.

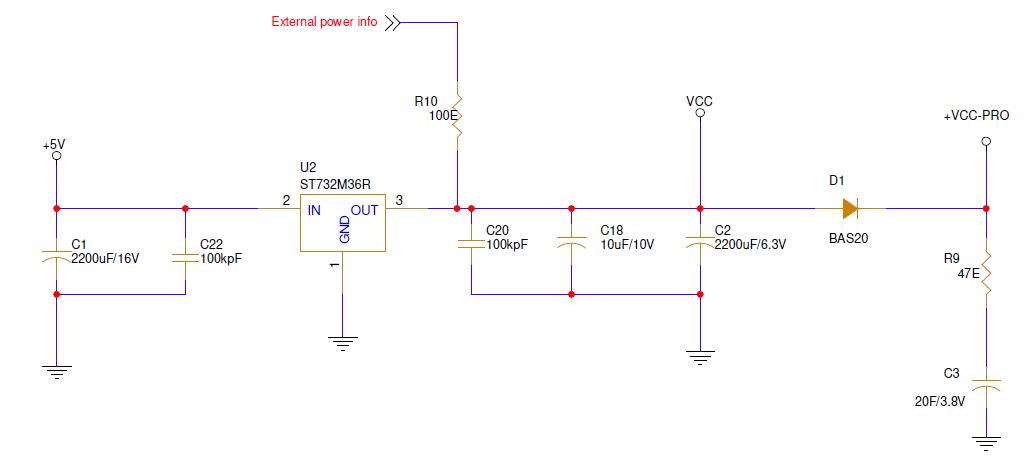
Acceptable Level:

MIN Voltage: +1.62V DC

MAX Voltage: + 3.6V DC

Current Output = 150mA

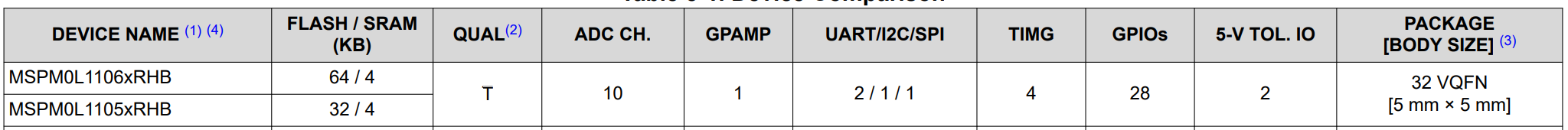
Measure across: C2



**2.1.2 Microcontroller:**

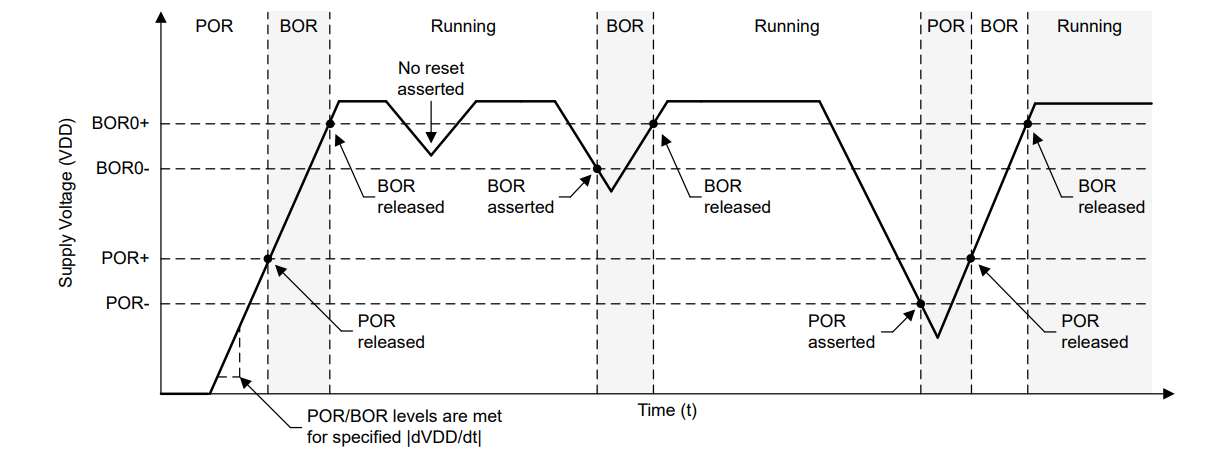
**Microcontroller Proposed:**

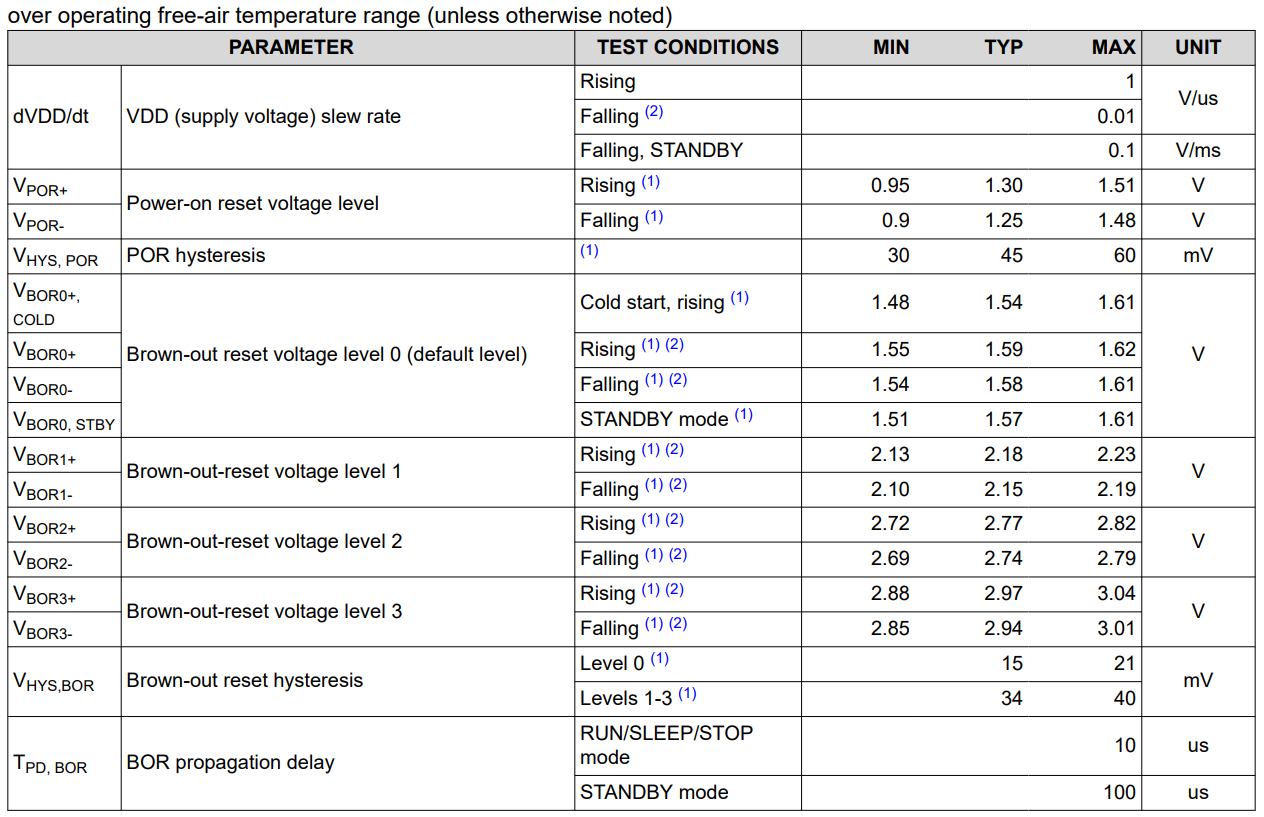
**MSPM0L1106TRHB (T = –40°C to 105°C)**

****

**Power on Reset and Brown out details:**

The below picture shows the relationships of POR-, POR+, BOR0-, and BOR0+ during powerup and power down.

****

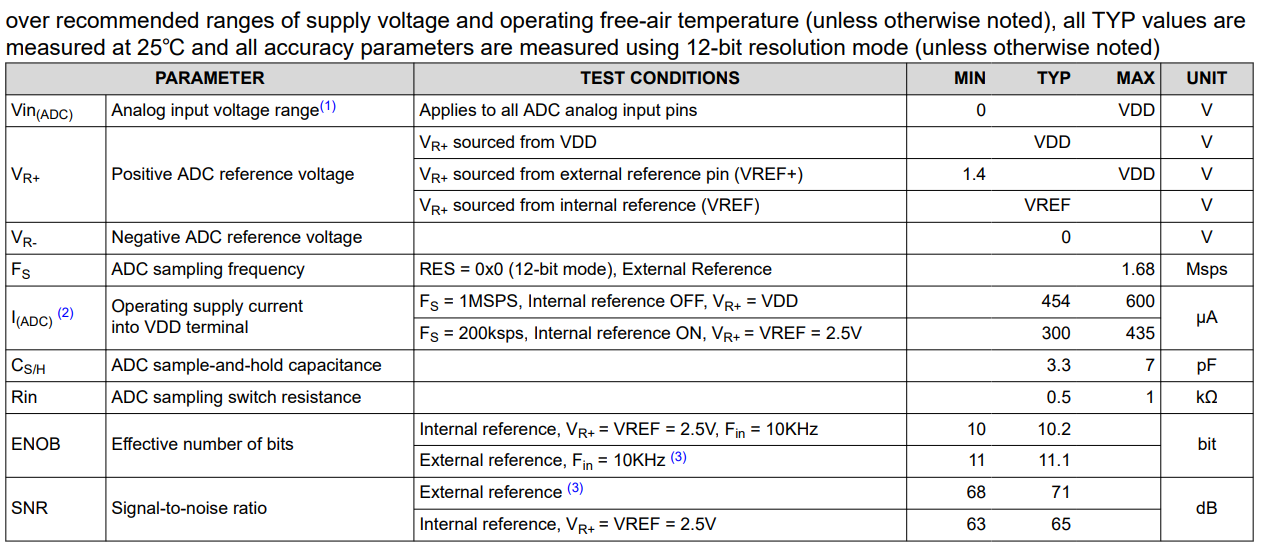
****

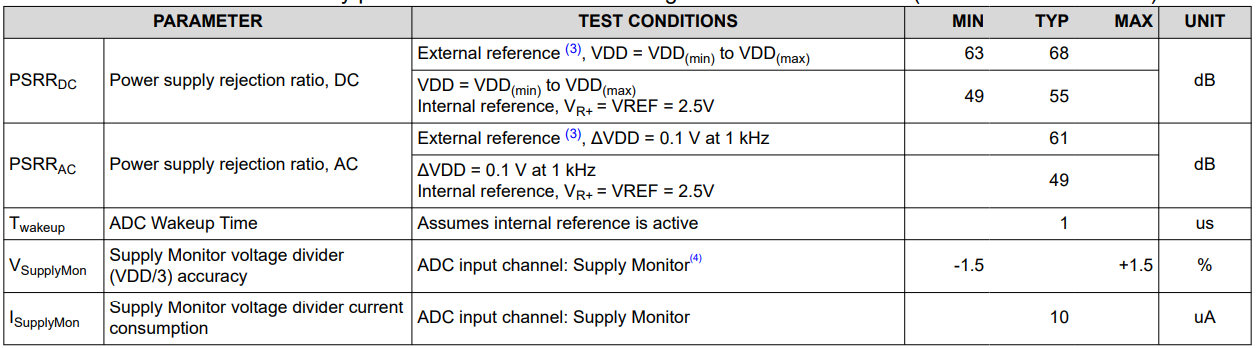
**2.1.3 Clock:**

The microcontroller is powered from the external power and supercapacitor / LIC capacitor and runs at 4MHz irrespective of external power or internal power

**2.1.4 Digital Inputs:**

All the DI inputs (DIO - DI5) are pulled up and the common connection will be ground, Normally the input is high and whenever the Potential free contact closes the GPIO will become Low. The raising and falling edge has to be detected and if the difference is valid count the pulse as valid pulse

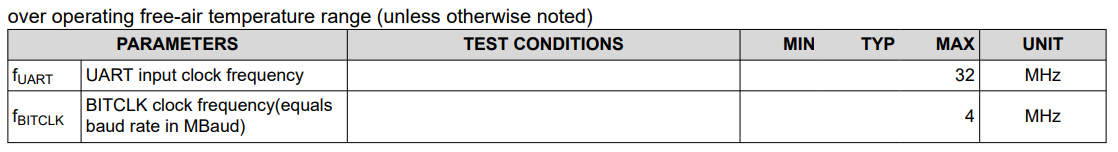
**** **2.1.4 ADC Electrical Characteristics:**

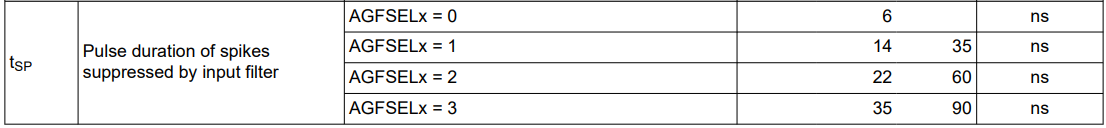
****

**2.1.4 External Interfaces:**

**UART:**

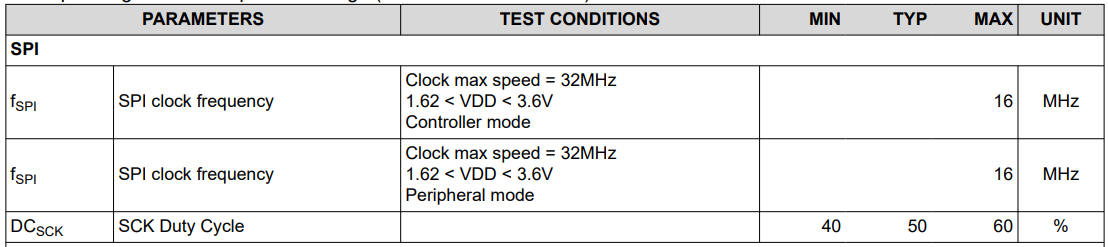
* Two UART ports are available (UART0 and UART1)



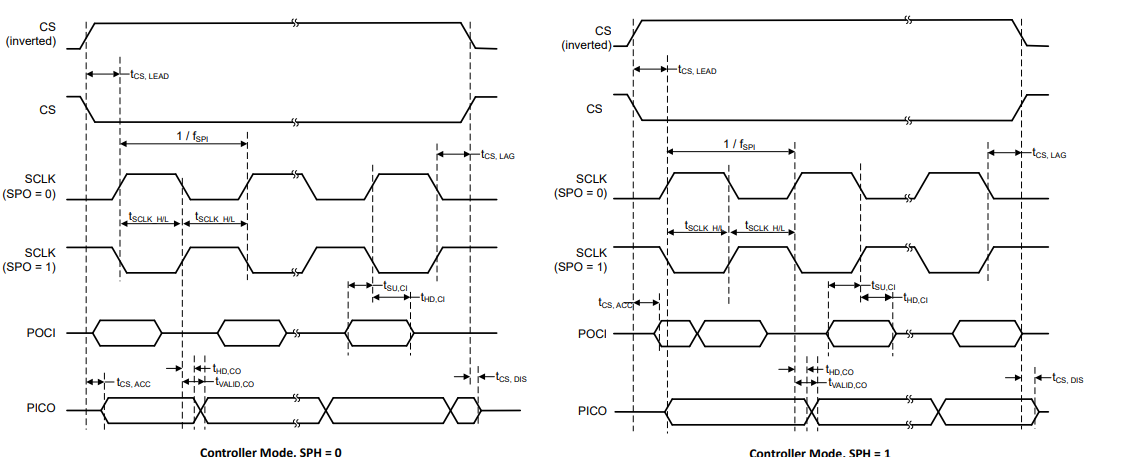
****

**SPI:**

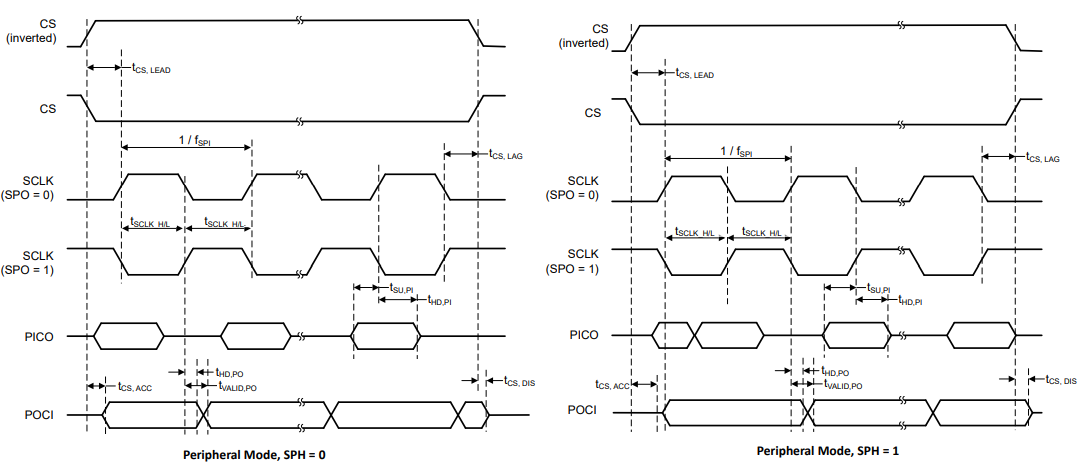
* One SPI port is available.



**SPI Timing Diagram: (Controller Mode)**

****

**SPI Timing Diagram: ( Peripheral Mode)**

****

## Architecture Hardware

## Architecture Firmware

The following peripheral has to be used in the design

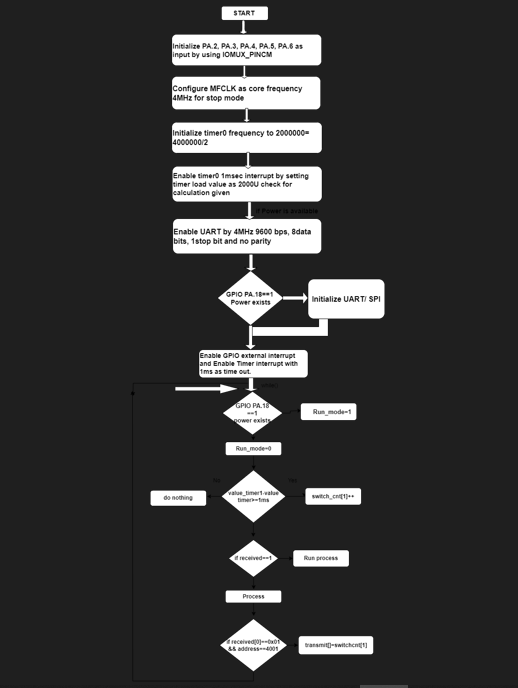
1. The internal Clock has to be initialized with 4MHz
2. Initialize Timer at 1000Hz
3. Initialize the GPIO single port- 6 pins with raising and falling edge
4. Initialize the GPIO pin to monitor external power (checking for the external power will be continuously done in while loop)
5. Initialize the UART if external power is available (before while loop need to check the external power and then initialize)
6. Initialize the SPI if external power is available (before while loop need to check the external power and then initialize)
7. Initialize the BOR 1 with the interrupt, once this interrupt occurs take the backup of data and the device goes to sleep mode continuously and only the clock and timer will be operational
8. Inside the interrupt of BOR1, initialize the interrupt at BOR0, and till BOR0 clock and Timer are working
9. **Once the Voltage reaches BOR0, the interrupt is generated. A trial needs to be done on how to reset the controller on the power resume**
10. Make sure that all the other peripherals are in the reset mode so that it cannot consume power
11. Whenever the raising and falling edge happens the GPIO interrupt (highest priority) is generated, raises the flag, and stores the present time. Once on the failing edge the interrupt is generated and the pulse width is calculated based on the timer. If the pulse is passing the minimum and maximum requirements, add the pulse.
12. The same can be communicated to the UART and SPI using the following Modbus protocol register map. These parameters can be readable individually and together

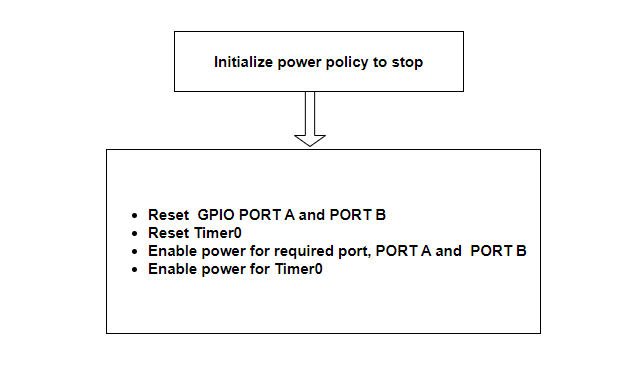
|  |  |  |  |
| --- | --- | --- | --- |
| **SL No** | **Parameter** | **Address** | **Data type** |
| 1 | Count Digital input 1 | 4001 | Unsigned long |
| 2 | Count Digital input 2 | 4003 | Unsigned long |
| 3 | Count Digital input 3 | 4005 | Unsigned long |
| 4 | Count Digital input 4 | 4007 | Unsigned long |
| 5 | Count Digital input 5 | 4009 | Unsigned long |
| 6 | Count Digital input 6 | 4011 | Unsigned long |
| 7 | Status Digital input 1 | 4013 | Unsigned long |
| 8 | Status Digital input 2 | 4015 | Unsigned long |
| 9 | Status Digital input 3 | 4017 | Unsigned long |
| 10 | Status Digital input 4 | 4019 | Unsigned long |
| 11 | Status Digital input 5 | 4021 | Unsigned long |
| 12 | Status Digital input 6 | 4023 | Unsigned long |

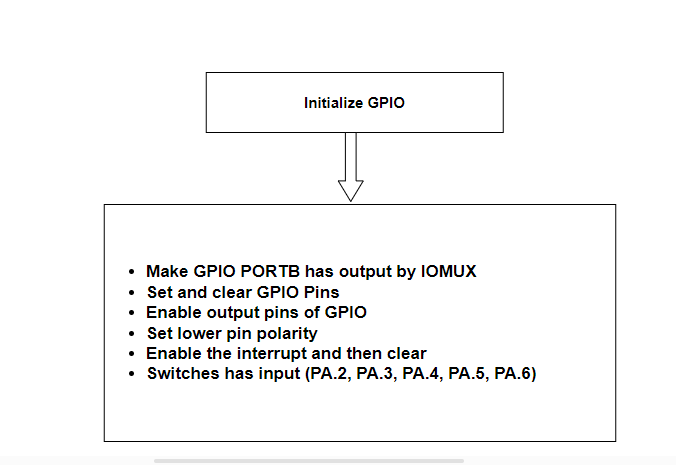
**Table 1**

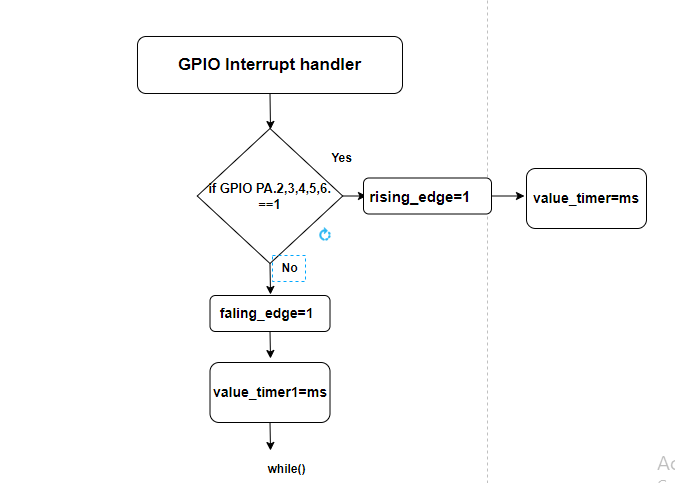
1. The following register should be present in the communication map for field debugging purposes together
   1. 6DI status
   2. 6DI ON time and OFF time
   3. How many times the individual DIs not passed the minimum and Max timing situation
   4. What was the last time the power occurred
   5. Need to maintain 1 and 20 registers including the running 1-second counter
   6. DI reset address and length for all 8 DI (6851\_ Write Single Coil (05))
   7. DI reset address and length for all individual DI
   8. In the DI communication counter can be add RPM of DI 1 to DI8 of last minute (Individually for DI1 to DI8)
   9. ON Hours Main Power
   10. ON Hours LIC Power
   11. Load Hours
   12. DI Counts force write for testing the DI auto reset. Address
   13. DI updating info status address for debug
   14. Meter info (address: 1, Length: 20)
   15. Write SA card SL No, Read SA card SL. No
   16. Write Meter SL No, Read Meter SL. No
   17. Calibration details (If any??)
   18. Dynamic communication parameter selection/write address
   19. Dynamic communication programmable Address
   20. Dynamic communication read address
   21. Address for selection for 4DI, 6DI, 8DI??? (For production option)
2. UART communication always works on a 9600 baud rate and even parity (None Parity?)
3. SPI communication will work with 250kHz(?) baud. To be discussed
4. LED to be blinking either for UART or for an SPI communication

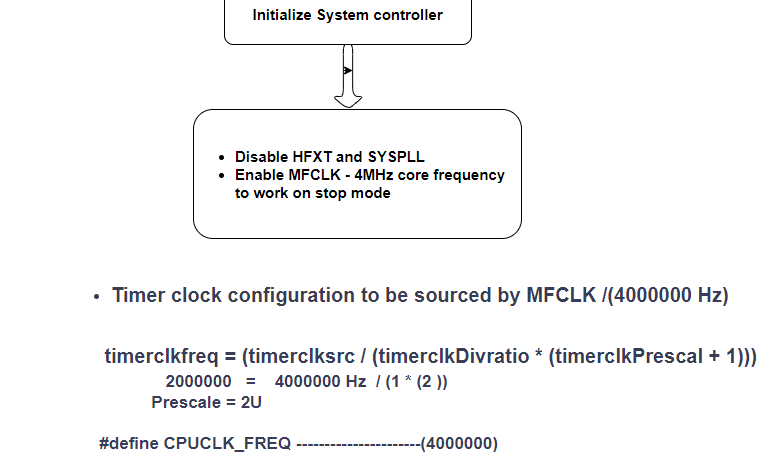
## 6. Flow chart :

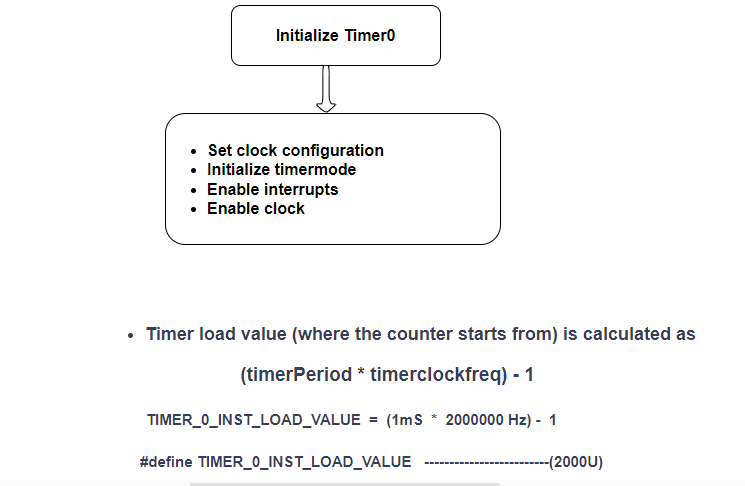


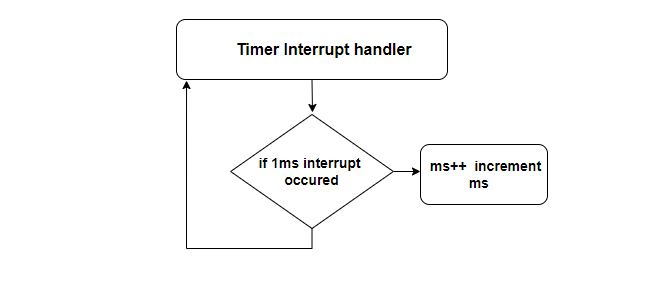


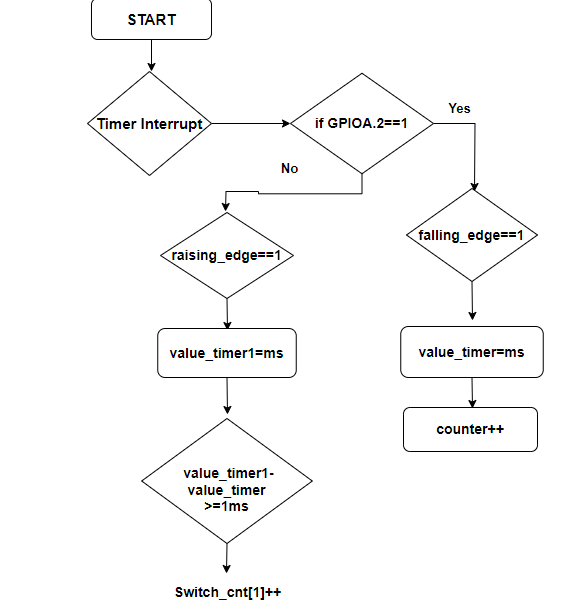












## 7. Algorithm :

void TIMER\_0\_INST\_IRQHandler(void)

{

static uint32\_t count = 0;

switch (DL\_TimerG\_getPendingInterrupt(TIMER\_0\_INST)) {

case DL\_TIMERG\_IIDX\_ZERO:

count++;

// delay\_cycles(1000000);

DL\_GPIO\_togglePins(GPIO\_LEDS\_PORT, GPIO\_LEDS\_USER\_LED\_2\_PIN);

data\_read = DL\_GPIO\_readPins(GPIOA, 0x3FFFC );

data\_read = data\_read >> 2;

for(int i=0;i<6;i++)

{

if(((data\_read & (0x0001 << i)) == 0)&& (El\_meter\_count\_falling\_Flag[i] == 1))

{

delay\_cycles(1000000);

DL\_GPIO\_togglePins(GPIO\_LEDS\_PORT, GPIO\_LEDS\_USER\_LED\_1\_PIN);

El\_meter\_count\_falling[i] = count;

El\_meter\_count\_falling\_Flag[i] = 0;

}

else if (((data\_read & (0x0001 << i)) != 0) /\*&& (El\_meter\_count\_falling\_Flag[i] == 1)\*/)

{

DL\_GPIO\_clearPins(GPIO\_LEDS\_PORT, GPIO\_LEDS\_USER\_LED\_1\_PIN);

El\_meter\_count\_falling\_Flag[i] = 1;

El\_meter\_count\_Raising[i] = count;

if(((El\_meter\_count\_Raising[i] - El\_meter\_count\_falling[i]) >= minimumCount) && ((El\_meter\_count\_Raising[i]- El\_meter\_count\_falling[i]) <= MaximumCount))

{

//delay\_cycles(1000000);

// DL\_GPIO\_setPins(GPIO\_LEDS\_PORT, GPIO\_LEDS\_USER\_LED\_1\_PIN);

}

}

if(entered\_BOR == 1)

{

/\* read GPIO pin for mains avilability \*/

if(mains\_avilable == 1)

{

Run\_Mode = 1;

}

else

{

Run\_Mode = 0;

}

}

}

break;

default:

break;

}

}

void NMI\_Handler(void)

{

switch (DL\_SYSCTL\_getRawNonMaskableInterruptStatus(DL\_SYSCTL\_NMI\_IIDX\_BORLVL))

{

case DL\_SYSCTL\_NMI\_IIDX\_BORLVL:

{

DL\_SYSCTL\_setBORThreshold(DL\_SYSCTL\_BOR\_THRESHOLD\_LEVEL\_0);

DL\_SYSCTL\_disableInterrupt( DL\_SYSCTL\_NMI\_BORLVL);

//NVIC\_DisableIRQ(NonMaskableInt\_VECn);

//flag=1;

entered\_BOR = 1;

DL\_SYSCTL\_clearInterruptStatus(DL\_SYSCTL\_NMI\_IIDX\_BORLVL);

//DL\_TimerG\_enableInterrupt(TIMER\_0\_INST , DL\_TIMERG\_INTERRUPT\_ZERO\_EVENT);

break;

}

default:

break;

}

}

void GROUP1\_IRQHandler(void)

{

/\*

\* Get the pending interrupt for the GPIOA port and store for

\* comparisons later

\*/

uint32\_t gpioA = DL\_GPIO\_getEnabledInterruptStatus(GPIOA,

GPIO\_SWITCHES\_USER\_SWITCH\_1\_PIN);

//switch (DL\_Interrupt\_getPendingGroup(DL\_INTERRUPT\_GROUP\_1)) {

//case GPIO\_SWITCHES\_GPIOA\_INT\_IIDX:

/\*

\* Bitwise AND the pending interrupt with the pin you want to check,

\* then check if it is equal to the pins. Clear the interrupt status.

\*/

if ((gpioA & GPIO\_SWITCHES\_USER\_SWITCH\_1\_PIN) == 1)

{

//delay\_cycles(10000000);

for(int i=0;i<826;i++)

{

delay\_cycles(1000000);

}

if (DL\_GPIO\_readPins(

GPIO\_SWITCHES\_USER\_SWITCH\_1\_PORT , GPIO\_SWITCHES\_USER\_SWITCH\_1\_PIN))

{

//delay\_cycles(1000000);

DL\_GPIO\_togglePins(GPIO\_LEDS\_PORT, GPIO\_LEDS\_USER\_LED\_1\_PIN); }

else{

//delay\_cycles(1000000);

DL\_GPIO\_clearPins(GPIO\_LEDS\_PORT, GPIO\_LEDS\_USER\_LED\_1\_PIN);

}

}

}