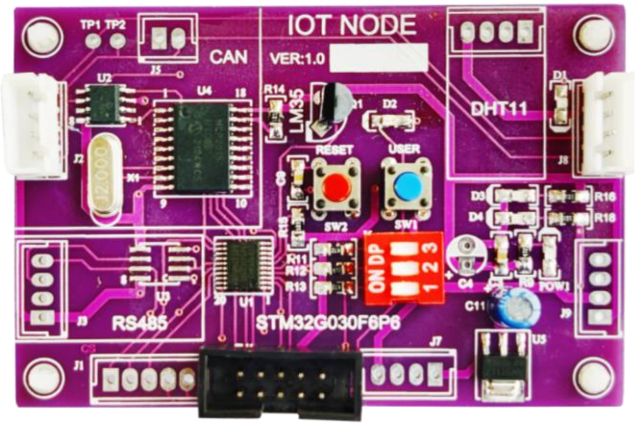
KERNEL MASTERS

CAN IoT Node

User Manual Rev 1.0/1.1



Contents

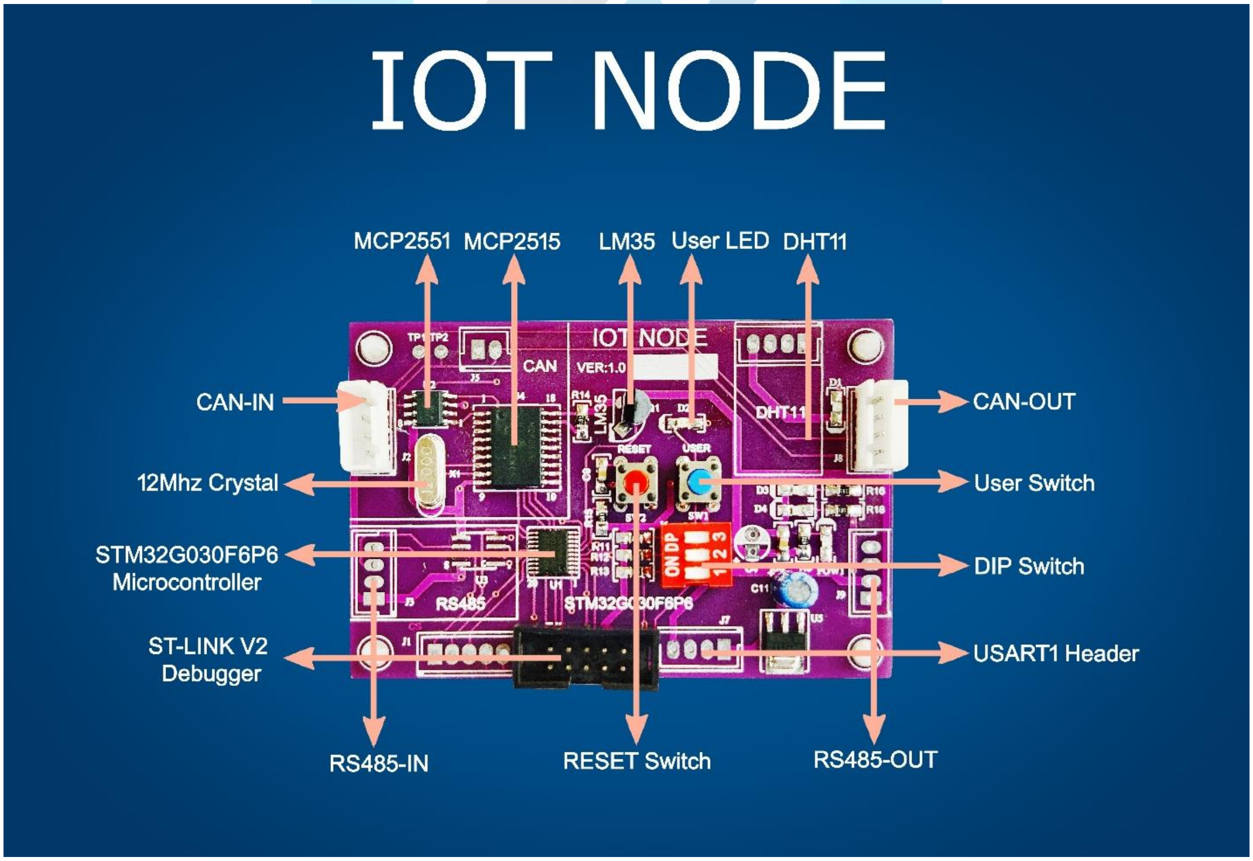
Chapter 1

1.1. BOARD OVERVIEW

The IoT Node board is generally used for wide range of IoT applications like automotive, environmental monitoring, industrial control etc. The IoT Node board includes the most common peripheral components for IoT solutions that gives unrivalled flexibility for developers building IoT (Internet of Things) nodes using CAN Bus.

The CAN IoT NODE is based on STM32G030F6P6 microcontroller that is based on high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at up to 64 MHz frequency. They are suitable for a wide range of applications in consumer, industrial and appliance domains and ready for the Internet of Things (IoT) solutions. This board also includes a CAN bus module which is controlled via SPI interface. It contains CAN Controller MCP2515 which is a high speed CAN trans-receiver. This module can be easily interfaced to any microcontroller via SPI interface.

Figure 1-1. CAN IoT Node Board

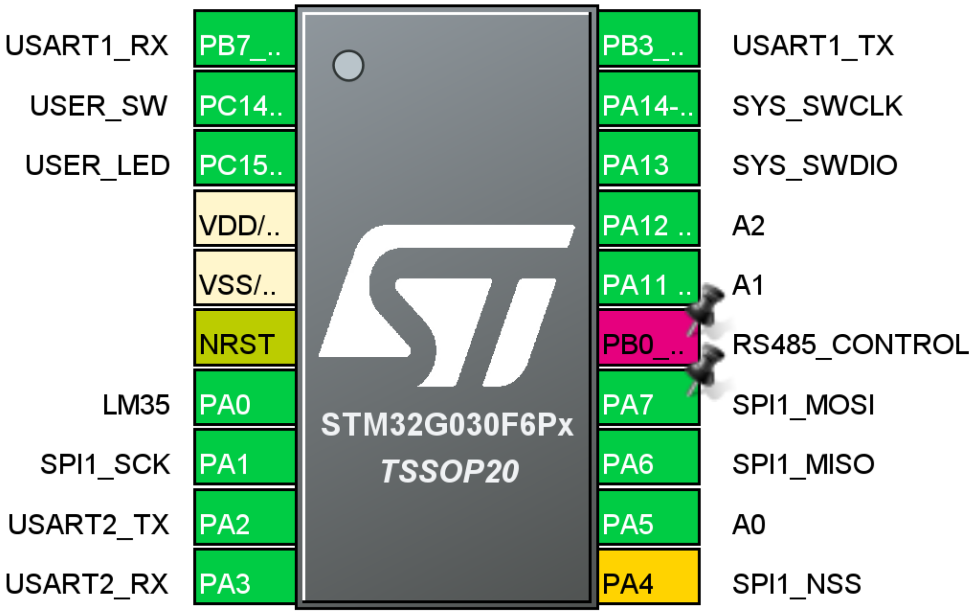


1.2. Features

- The STM32G030F6P6 microcontroller based on high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at up to 64 MHz frequency. It incorporate high-speed embedded memories (8 Kbytes of SRAM and up to 64 Kbytes of Flash program memory with read protection, write protection).
- On-board ST-LINK Serial wire debugger/programmer.
- On-board CAN Controller with SPI Interface.
- RS485 Serial communication interface.
- DHT11 interface available.
- Dip switch to set the corresponding node id.
  - can set any number between 0 to 7
- On – board temperature sensor LM35.
- User (application) and reset push-buttons.
- Power LEDs and a user (Green) LED.
- STM32G030x6/x8 devices require a 2.0 V to 3.6 V operating supply voltage (VDD).
- Support of a wide choice of Integrated Development Environments (IDEs) including IAR Embedded Workbench®, MDK-ARM, and STM32CubeIDE.

1.3. IoT Node Pinout

Figure 1-2 IoT Node Pinout



1.4. Specifications

Table 1-1 summarizes the specifications for the IoT Node board.

Table 1-1. IoT Node Board Specifications

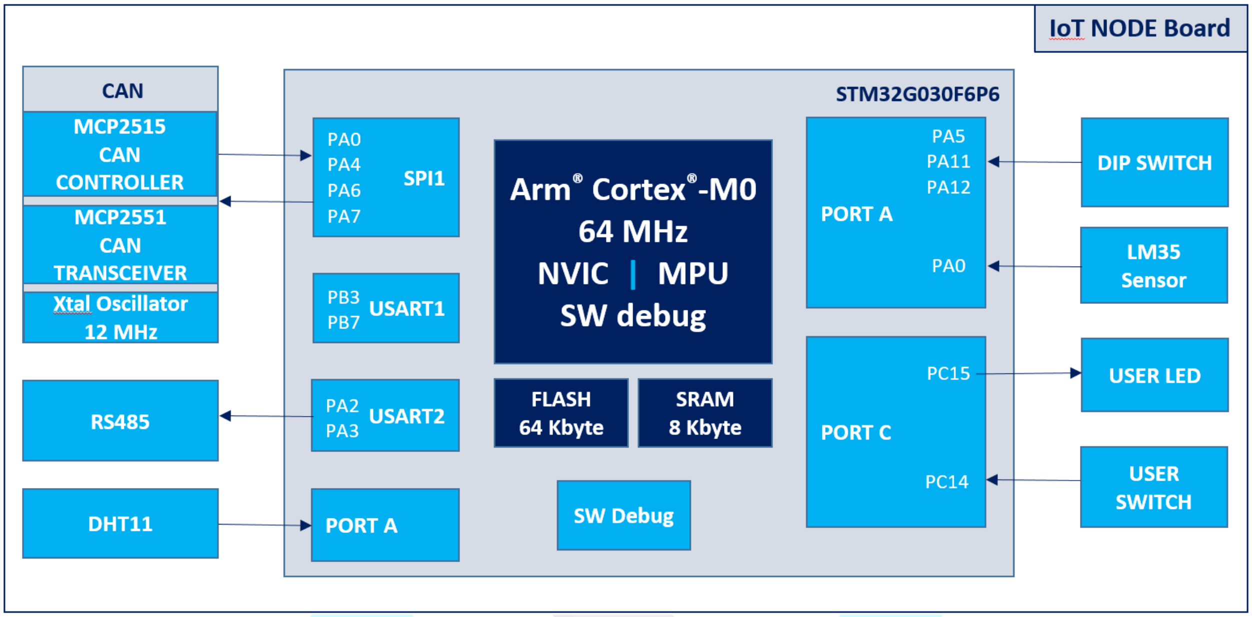
Parameter	**Value**
Board supply voltage	2.0 V to 5.5 V
Dimensions	(80 x 53 x 10)
Break-out power output	• 3.3 VDC (current rating) • 5.0 VDC (current rating)
RoHS status	Compliant

Chapter 2

2.1. MICROCONTROLLER HARDWARE DESCRIPTION

The STM32G030F6P6 based IoT Node board includes few hardware components, an integrated serial wire debugger as well as a range of useful peripherals.

Figure 2-1. IoT Node Board Block Diagram

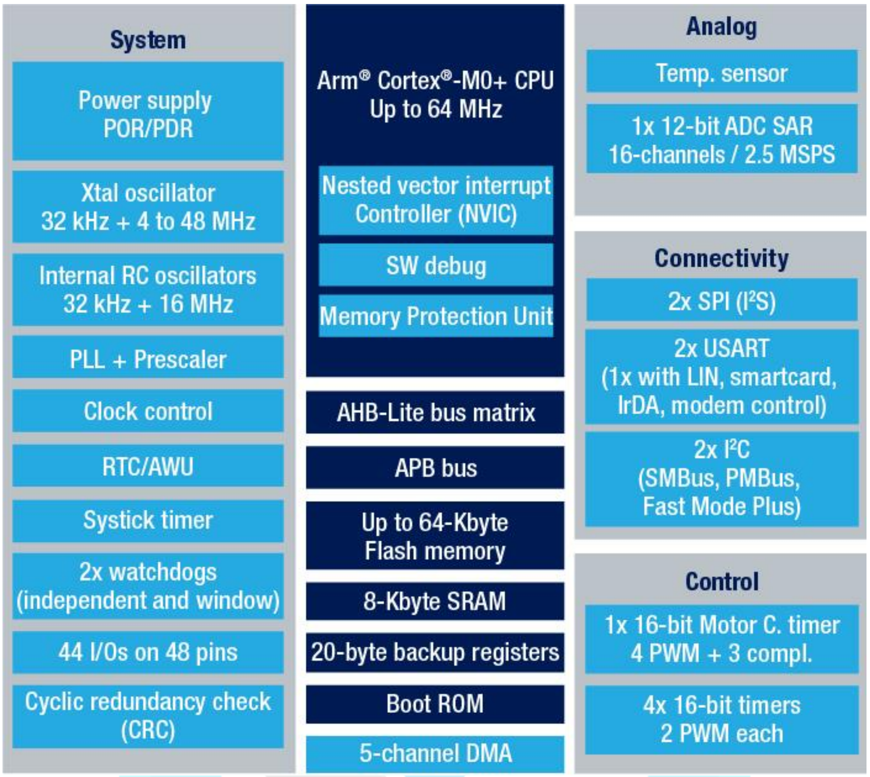


2.2. Microcontroller

The STM32G030F6P6 is a mainstream microcontroller which is based on high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at up to 64 MHz frequency. The devices incorporate a memory protection unit (MPU), high-speed embedded memories (8 Kbytes of SRAM and up to 64 Kbytes of Flash program memory with read protection, write protection), DMA, an extensive range of system functions, enhanced I/Os, and peripherals (*See the STM32F401RBT6 microcontroller data sheet*).

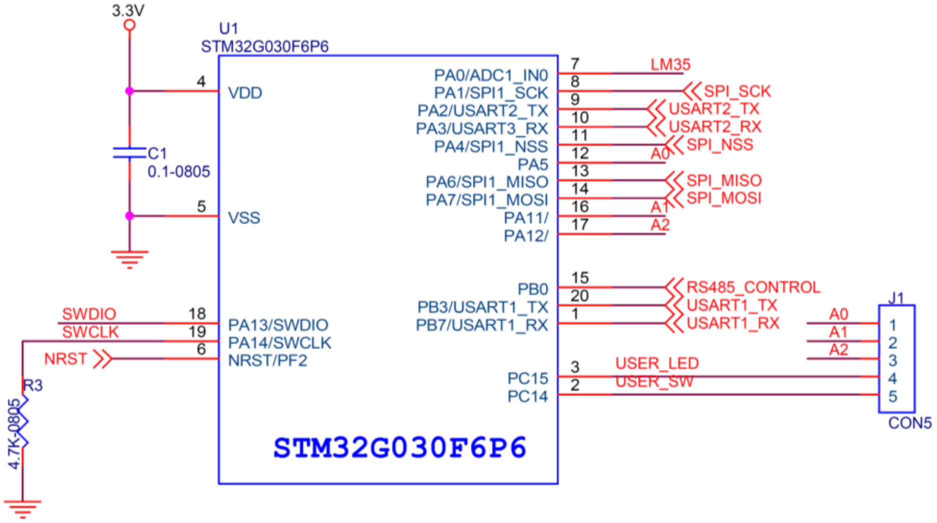
The devices offer standard communication interfaces (two I^2Cs, two SPIs / one I^2S, and two USARTs), one 12-bit ADC (2.5 MSps) with up to 19 channels, a low-power RTC, an advanced control PWM timer, four general-purpose 16-bit timers, two watchdog timers, and a SysTick timer. The devices operate within ambient temperatures from -40 to 85°C and with supply voltages from 2.0 V to 3.6 V.

Figure 2-2. STM32G030F6P6 Block Diagram


System
Power supply POR/PDR
Xtal oscillator 32 kHz + 4 to 48 MHz
Internal RC oscillators 32 kHz + 16 MHz
PLL + Prescaler
Clock control
RTC/AWU
Systick timer
2x watchdogs (independent and window)
44 I/Os on 48 pins
Cyclic redundancy check (CRC)
Arm® Cortex®-M0+ CPU Up to 64 MHz
Nested vector interrupt Controller (NVIC)
SW debug
Memory Protection Unit
AHB-Lite bus matrix
APB bus
Up to 64-Kbyte Flash memory
8-Kbyte SRAM
20-byte backup registers
Boot ROM
5-channel DMA
Analog
Temp. sensor
1x 12-bit ADC SAR 16-channels / 2.5 MSPS
Connectivity
2x SPI (I²S)
2x USART (1x with LIN, smartcard, IrDA, modem control)
2x I²C (SMBus, PMBus, Fast Mode Plus)
Control
1x 16-bit Motor C. timer 4 PWM + 3 compl.
4x 16-bit timers 2 PWM each

The devices come in packages with 8 to 48 pins. Figure 2-3 shows the schematic diagram of how the microcontroller is interfaced with other devices on board.

Figure 2-3 Schematic – Microcontroller



2.3. Switches and LEDs

The IoT Node board has included 4 LEDs out of which 3 are power LEDs and the remaining one is a user LED. It also has 2 switches, out of which 1 is reset button and the other is a user switch. The user switch and/or LED can be used for serving some purposes in the custom applications. The user switch is mentioned on the board as SW1, and the user LED is represented on the board as D2, which is a green LED. The pin details regarding the switch and LED is given in Table 1-1. The schematic are also given below.

Figure 2-4 Schematic - LEDs and Switches

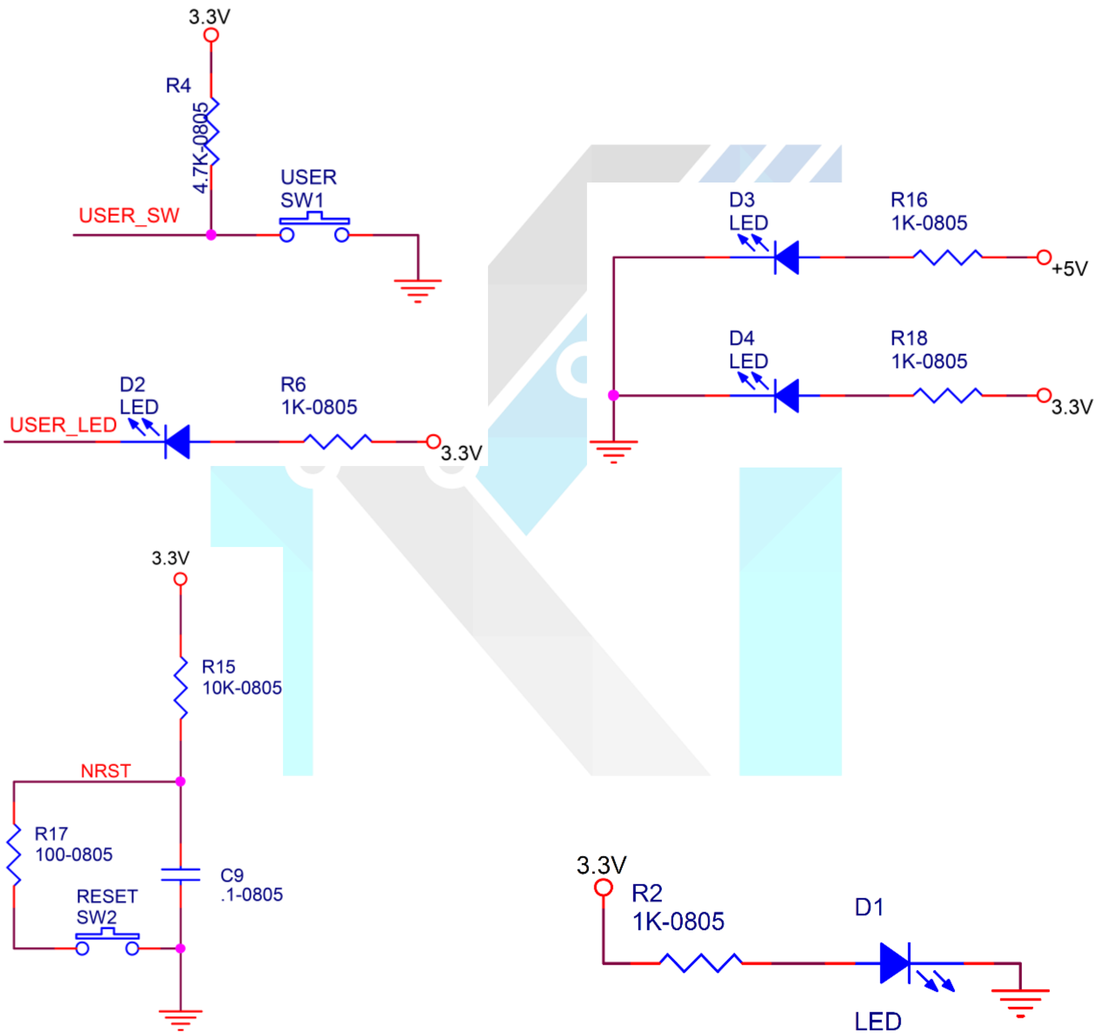


Table 1-1 Pin Description for User switch and LED

GPIO PIN	Pin Function	On-board Function
PC14	GPIO	USER_SW (SW1)
PC15	GPIO	USER_LED (Green)

2.4. DIP Switch

A DIP switch is a dual in-line package switch, meaning that it consists of a series of switches in a single unit. Here, the IoT Node board has a 3 way DIP switch to set the node ID for each node in case of multiple IoT nodes. The dip switch has 3 switches and the ID is read as per the switch configuration. The node ID ranges from 0 to 7. The node ID is the identifier for each node wherever it is used. Table 1-2 shows how to configure the switches to select node IDs.

Figure 2-5 Schematic - DIP Switches

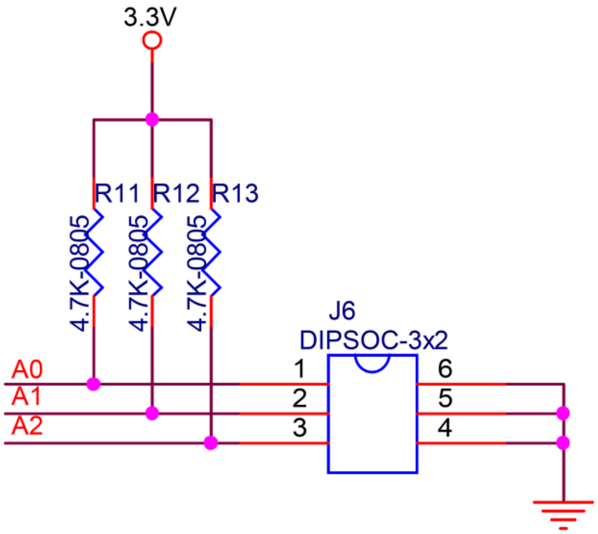


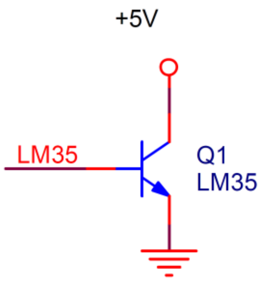
Table 1-2 DIP switch Configuration for Node ID

A2 (1)	A1 (2)	A0 (3)	NODE ID
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

2.5. LM35 Temperature Sensor

The IoT Node board has an on board LM35 temperature sensor which is connected to pin **PA0**. It is internally connected to ADC1 channel 0 (ADC1_IN0). The ADC will convert the analog output from LM35 to its corresponding digital value.

Figure 2-6 Schematic - LM35



2.6. ST-Link V2 Debugger/Programmer

The ST-LINK/V2 is an in-circuit debugger and programmer for the STM8 and STM32 microcontrollers. The IoT Node board uses serial wire debugging (SWD) interface to communicate with the STM32 microcontroller located on the development board. The SWD can be connected to CN1 connector on the board. Table 1-3 shows the pin description for Debugger/Programmer.

Figure 2-7 Schematic – ST-Link V2 Debugger/Programmer

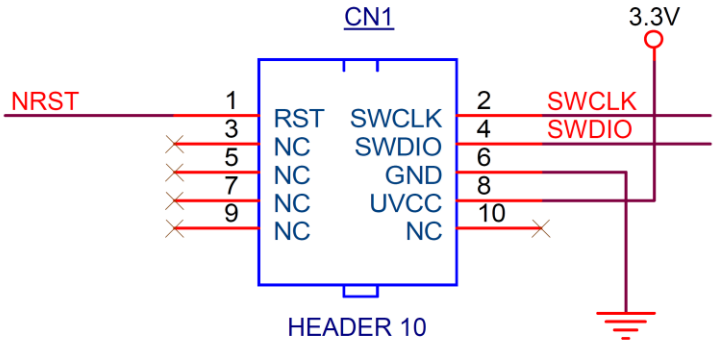
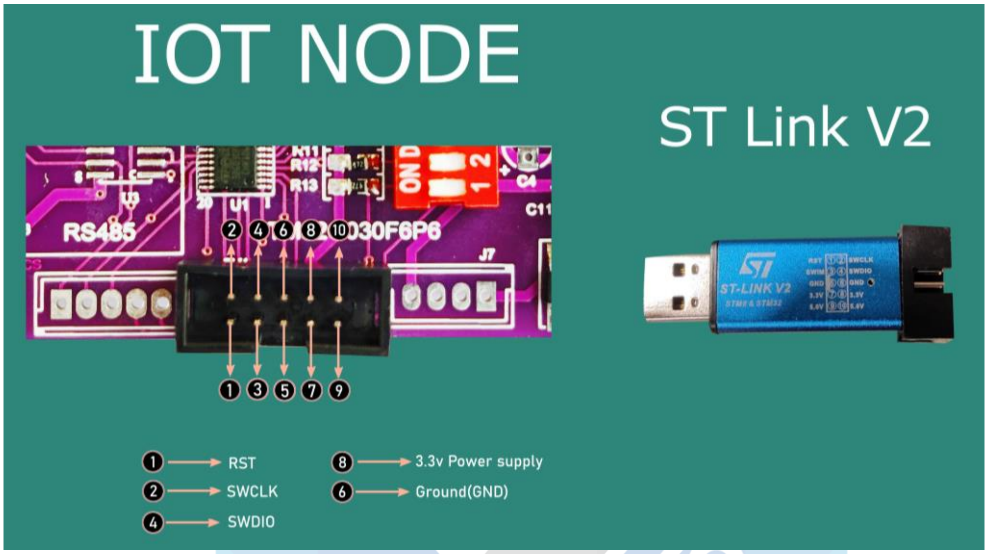


Table 1-3 Pin Description for ST-Link V2 Debugger

GPIO PIN	PIN FUNCTION	DEVICE
PA13	GPIO	SWDIO
PA14	GPIO	SWCLK
PF2	GPIO	NRST



2.7. RS485 zone

RS-485 is an industrial specification that defines the electrical interface and physical layer for point-to-point communication of electrical devices. The RS-485 standard allows for long cabling distances in electrically noisy environments and can support multiple devices on the same bus.

Figure 2-8 Schematic – RS485

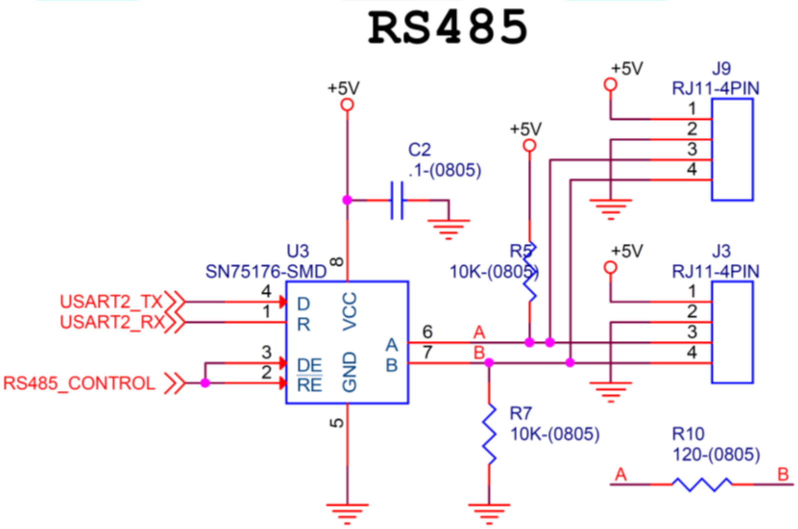


Table 1-4 Pin Description for RS485

GPIO PIN	PIN FUNCTION	DEVICE
PA2	GPIO	USART2_TX
PA3	GPIO	USART2_RX
PB0	GPIO	RS485_CONTROL

2.8. USART zone

UART, or universal asynchronous receiver-transmitter, is one of the most used device-to-device communication protocols. When properly configured, UART can work with many different types of serial protocols that involve transmitting and receiving serial data. The IoT Node Board provides a dedicated UART header which helps in communicating various peripherals, sensors etc. The UART header is named J7, which is connected to UART1. Table 1-5 shows the pin description for USART.

Figure 2-9 Schematic – USART

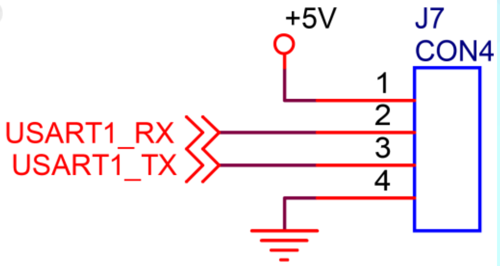


Table 1-5 Pin Description for USART

GPIO PIN	PIN FUNCTION	DEVICE
PB3	GPIO	USART1_TX
PB7	GPIO	USART1_RX

Chapter 3

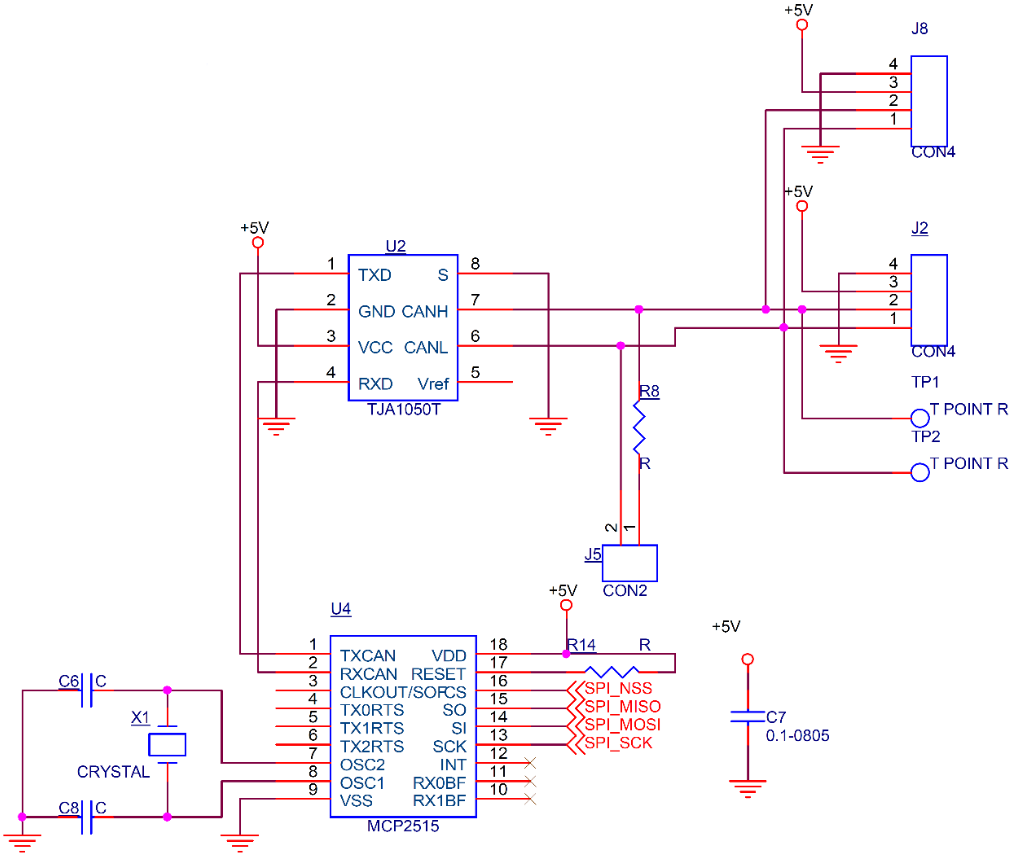
3.1. CAN CONTROLLER HARDWARE DESCRIPTION

The Controller Area Network (CAN) is a serial, asynchronous, multi-master communication protocol for connecting electronic Control modules, sensors and actuators in automotive and industrial applications. The CAN Controller is an interface between the Application and the CAN bus. The function of the CAN Controller is to convert the data provided by the application into a CAN message frame fit to be transmitted across the bus. There are mainly three hardware components associated with CAN controller.

- MCP2515 CAN Controller
- MCP2551 CAN Transceiver
- Oscillator

Figure 3-1 shows the schematic diagram of how CAN Controller is interfaced on board with the STM32G030F6P6 microcontroller.

Figure 3-1 Schematics – CAN Controller



2. CONTROLLER AREA NETWORK (CAN)

2.1. MCP2515 CAN CONTROLLER

MCP2515 is a stand-alone Controller Area Network (CAN) controller that implements the CAN specification, Version 2.0B. It is capable of transmitting and receiving both standard and extended data and remote frames. The MCP2515 has two acceptance masks and six acceptance filters that are used to filter out unwanted messages, thereby reducing the host MCU's overhead. The MCP2515 interfaces with microcontrollers (MCUs) via an industry standard Serial Peripheral Interface (SPI).

2.1.1. Features

- Implements CAN V2.0B at 1 Mb/s:
  - 0 to 8-byte length in the data field
  - Standard and extended data and remote frames
- Receive Buffers, Masks and Filters:
  - Two receive buffers with prioritized message storage
  - Six 29-bit filters
  - Two 29-bit masks
- Data Byte Filtering on the First Two Data Bytes (applies to standard data frames)
- Three Transmit Buffers with Prioritization and Abort Features
- High-Speed SPI Interface (10 MHz):
  - SPI modes 0,0 and 1,1
- One-Shot mode Ensures Message Transmission is Attempted Only One Time
- Clock Out Pin with Programmable Prescaler:
  - Can be used as a clock source for other device(s)
- Start-of-Frame (SOF) Signal is Available for Monitoring the SOF Signal:
  - Can be used for time slot-based protocols and/or bus diagnostics to detect early bus degradation.
- Interrupt Output Pin with Selectable Enables
- Buffer Full Output Pins Configurable as:
  - Interrupt output for each receive buffer
  - General purpose output
- Request-to-Send (RTS) Input Pins Individually Configurable as:
  - Control pins to request transmission for each transmit buffer.
  - General purpose inputs
- Low-Power CMOS Technology:
  - Operates from 2.7V-5.5V
  - 5 mA active current (typical)
  - 1 µA standby current (typical) (Sleep mode)
- Temperature Ranges Supported:
  - Industrial (I): -40°C to +85°C

- Extended (E): -40°C to +125°C

2.1.2. *Pin out*

Figure 3-2 shows the pin out diagram for MCP2515. An 18 – Lead PDIP/SOIC package of the IC is used on the board. Also the Table 3-1 shows the pin out description.

Figure 3-2 MCP2515 Pin out Diagram

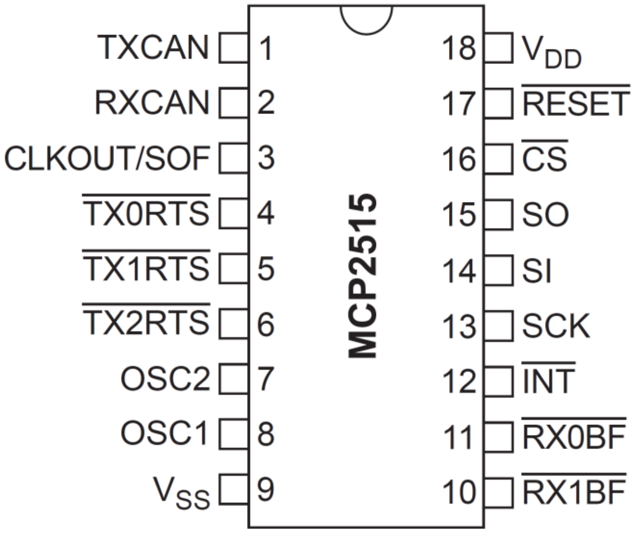
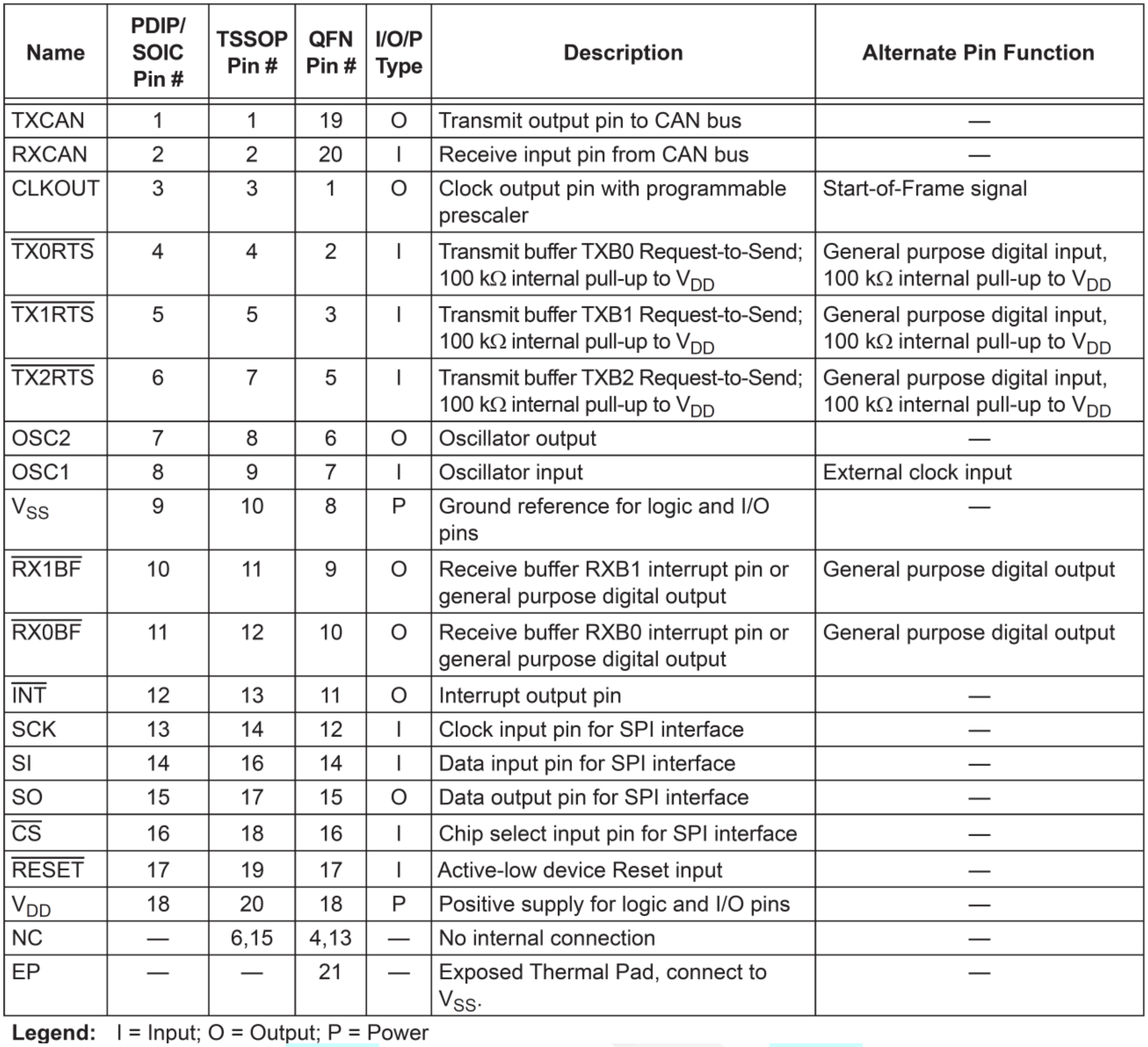


Table 3-1 Pin out Description



Name	PDIP/ SOIC Pin #	TSSOP Pin #	QFN Pin #	I/O/P Type	Description	Alternate Pin Function
TXCAN	1	1	19	O	Transmit output pin to CAN bus	—
RXCAN	2	2	20	I	Receive input pin from CAN bus	—
CLKOUT	3	3	1	O	Clock output pin with programmable prescaler	Start-of-Frame signal
$\overline{\text{TX0RTS}}$	4	4	2	I	Transmit buffer TXB0 Request-to-Send; 100 kΩ internal pull-up to V_{DD}	General purpose digital input, 100 kΩ internal pull-up to V_{DD}
$\overline{\text{TX1RTS}}$	5	5	3	I	Transmit buffer TXB1 Request-to-Send; 100 kΩ internal pull-up to V_{DD}	General purpose digital input, 100 kΩ internal pull-up to V_{DD}
$\overline{\text{TX2RTS}}$	6	7	5	I	Transmit buffer TXB2 Request-to-Send; 100 kΩ internal pull-up to V_{DD}	General purpose digital input, 100 kΩ internal pull-up to V_{DD}
OSC2	7	8	6	O	Oscillator output	—
OSC1	8	9	7	I	Oscillator input	External clock input
V_{SS}	9	10	8	P	Ground reference for logic and I/O pins	—
$\overline{\text{RX1BF}}$	10	11	9	O	Receive buffer RXB1 interrupt pin or general purpose digital output	General purpose digital output
$\overline{\text{RX0BF}}$	11	12	10	O	Receive buffer RXB0 interrupt pin or general purpose digital output	General purpose digital output
$\overline{\text{INT}}$	12	13	11	O	Interrupt output pin	—
SCK	13	14	12	I	Clock input pin for SPI interface	—
SI	14	16	14	I	Data input pin for SPI interface	—
SO	15	17	15	O	Data output pin for SPI interface	—
$\overline{\text{CS}}$	16	18	16	I	Chip select input pin for SPI interface	—
$\overline{\text{RESET}}$	17	19	17	I	Active-low device Reset input	—
V_{DD}	18	20	18	P	Positive supply for logic and I/O pins	—
NC	—	6,15	4,13	—	No internal connection	—
EP	—	—	21	—	Exposed Thermal Pad, connect to V_{SS}.	—

Legend: I = Input; O = Output; P = Power

2.1.3. Device Overview

The device consists of three main blocks:

1. The CAN module, which includes the CAN protocol engine, masks, filters, transmit and receive buffers.
2. The control logic and registers that are used to configure the device and its operation.
3. The SPI protocol block.

2.1.3.1. CAN Module

The CAN module handles all functions for receiving and transmitting messages on the CAN bus. Messages are transmitted by first loading the appropriate message buffer and control registers. Transmission is initiated by using control register bits via the SPI interface or by using the transmit enable pins. Status and errors can be checked by reading the appropriate registers. Any message detected on the CAN bus is checked for errors and then matched against the user-defined filters to see if it should be moved into one of the two receive buffers.

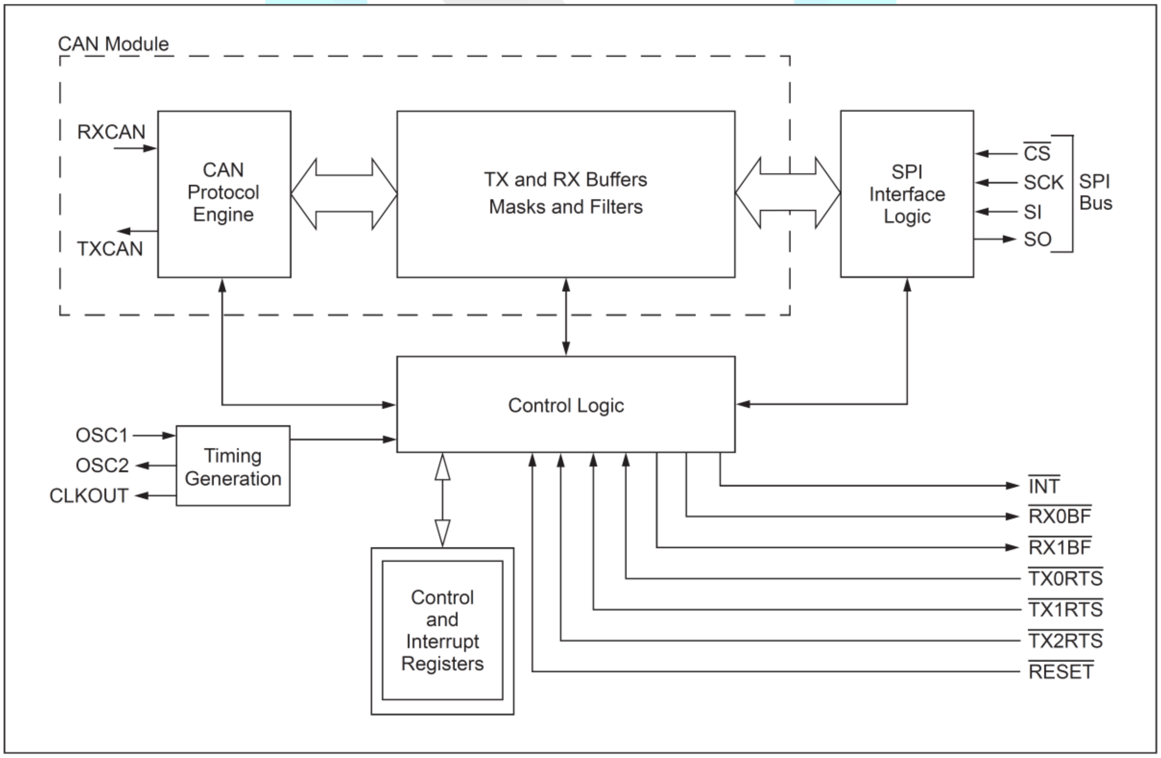
2.1.3.2. Control Logic

The control logic block controls the setup and operation of the MCP2515 by interfacing to the other blocks in order to pass information and control. Interrupt pins are provided to allow greater system flexibility. The general purpose interrupt pin, as well as status registers (accessed via the SPI interface), can also be used to determine when a valid message has been received. Additionally, there are three pins available to initiate immediate transmission of a message that has been loaded into one of the three transmit registers. Use of these pins is optional, as initiating message transmissions can also be accomplished by utilizing control registers accessed via the SPI interface.

2.1.3.3. SPI Protocol Block

The MCU interfaces to the device via the SPI interface. Writing to, and reading from, all registers is accomplished using standard SPI read and write commands, in addition to specialized SPI commands.

Figure 3-3 MCP2515 Block Diagram



2.1.4. *CAN MESSAGE FRAMES*

The MCP2515 supports standard data frames, extended data frames and remote frames (standard and extended), as defined in the CAN 2.0B specification.

2.1.4.1. Standard Data Frame

The frame begins with a Start-of-Frame (SOF) bit, which is of the dominant state and allows hard synchronization of all nodes. The SOF is followed by the arbitration field, consisting of 12 bits: the 11-bit identifier and the Remote Transmission Request (RTR) bit. The RTR bit is used to distinguish a data frame (RTR bit dominant) from a remote frame (RTR bit recessive).

Following the arbitration field is the control field, consisting of six bits. The first bit of this field is the Identifier Extension (IDE) bit, which must be dominant to specify a standard frame. The following bit, Reserved Bit Zero (RB0), is reserved and is defined as a dominant bit by the CAN protocol. The remaining four bits of the control field are the Data Length Code (DLC), which specifies the number of bytes of data (0-8 bytes) contained in the message.

After the control field, is the data field, which contains any data bytes that are being sent, and is of the length defined by the DLC (0-8 bytes). The Cyclic Redundancy Check (CRC) field follows the data field and is used to detect transmission errors. The CRC field consists of a 15-bit CRC sequence, followed by the recessive CRC Delimiter bit. The final field is the two-bit Acknowledge (ACK) field. During the ACK Slot bit, the transmitting node sends out a recessive bit.

Any node that has received an error-free frame Acknowledges the correct reception of the frame by sending back a dominant bit (regardless of whether the node is configured to accept that specific message or not). The recessive Acknowledge delimiter completes the Acknowledge field and may not be overwritten by a dominant bit. Please refer figure 3-3 (a).

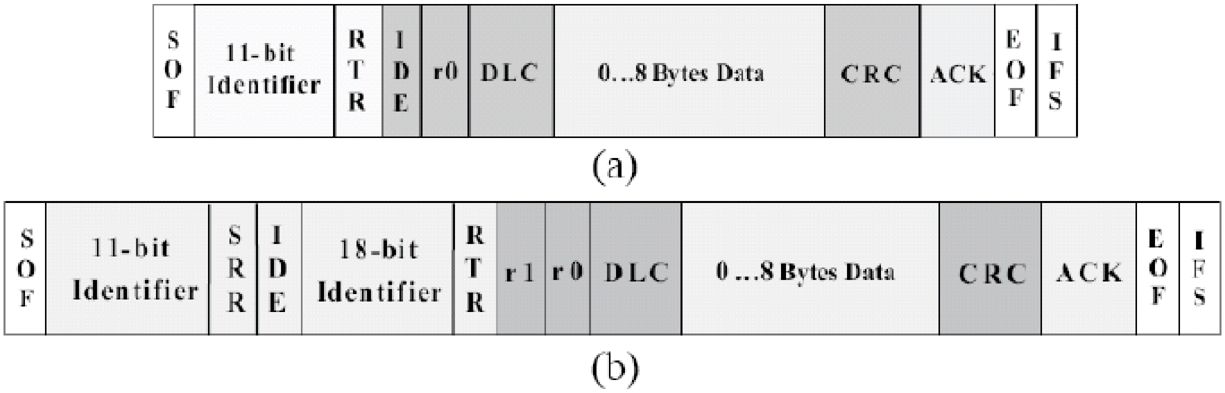
2.1.4.2. Extended Data Frame

In the extended CAN data frame, the SOF bit is followed by the arbitration field, which consists of 32 bits. The first 11 bits are the Most Significant bits (MSb) (Base-ID) of the 29-bit identifier. These 11 bits are followed by the Substitute Remote Request (SRR) bit, which is defined to be recessive.

The SRR bit is followed by the IDE bit, which is recessive to denote an extended CAN frame. It should be noted that if arbitration remains unresolved after transmission of the first 11 bits of the identifier, and one of the nodes involved in the arbitration is sending a standard CAN frame (11-bit identifier), the standard CAN frame will win arbitration due to the assertion of a dominant IDE bit. Also, the SRR bit in an extended CAN frame must be recessive to allow the assertion of a dominant RTR bit by a node that is sending a standard CAN remote frame. The SRR and IDE bits are followed by the remaining 18 bits of the identifier (Extended ID) and the Remote Transmission Request bit. To enable standard and extended frames to be sent across a shared network, the 29-bit extended message identifier is split into 11-bit (Most Significant) and 18-bit (Least Significant) sections.

This split ensures that the IDE bit can remain at the same bit position in both the standard and extended frames. Following the arbitration field is the six-bit control field. The first two bits of this field are reserved and must be dominant. The remaining four bits of the control field are the DLC, which specifies the number of data bytes contained in the message. The remaining portion of the frame (data field, CRC field, Acknowledge field, End-of-Frame and intermission) is constructed in the same way as a standard data frame. Please refer figure 3-3 (b).

Figure 3-4 (a) Standard CAN Frame and (b) Extended CAN Frame



(a)

(b)

2.1.5. Modes Of Operation

The MCP2515 has five modes of operation. These modes are:

1. Configuration mode
2. Normal mode
3. Sleep mode
4. Listen-Only mode
5. Loopback mode

The operational mode is selected via the REQOP [2:0] bits (CANCTRL [7:5]). When changing modes, the mode will not actually change until all pending message transmissions are complete. The requested mode must be verified by reading the OPMODE [2:0] bits CANSTAT [7:5].

2.1.5.1. Configuration Mode

The MCP2515 must be initialized before activation. This is only possible if the device is in the Configuration mode. Configuration mode is automatically selected after power-up, a Reset or can be entered from any other mode by setting the REQOP [2:0] bits to '100'. When Configuration mode is entered, all error counters are cleared. Configuration mode is the only mode where the following registers are modifiable:

- CNF1, CNF2, CNF3 registers
- TXRTSCTRL register
- Filter registers
- Mask registers

2.1.5.2. Sleep Mode

The MCP2515 has an internal Sleep mode that is used to minimize the current consumption of the device. The SPI interface remains active for reading even when the MCP2515 is in Sleep mode, allowing access to all registers. To enter Sleep mode, the Request Operation Mode bits are set in the CANCTRL register (REQOP [2:0]). The OPMODE [2:0] bits (CANSTAT [7:5]) indicate the operation mode. The MCP2515 is active and has not yet entered Sleep mode until these bits indicate that Sleep mode has been entered. When in internal Sleep mode, the wake-up interrupt is still active (if enabled). This is done so that the MCU can also be placed into a Sleep mode and use the MCP2515 to wake it up upon detecting activity on the bus.

When in Sleep mode, the MCP2515 stops its internal oscillator. The MCP2515 will wake-up when bus activity occurs or when the MCU sets, via the SPI interface, the
WAKIF bit (CANINTF [6]). To 'generate' a wake-up attempt, the WAKIE bit (CANINTE[6]) must also be set in order for the wake-up interrupt to occur. The TXCAN pin will remain in the recessive state while the MCP2515 is in Sleep mode.

2.1.5.3. Listen-Only Mode

Listen-Only mode provides a means for the MCP2515 to receive all messages (including messages with errors) by configuring the RXM [1:0] bits (RXBnCTRL[6:5]). This mode can be used for bus monitor applications or for detecting the baud rate in 'hot plugging' situations. Listen-Only mode is a silent mode, meaning no messages will be transmitted while in this mode (including error flags or Acknowledge signals). In Listen-Only mode, both valid and invalid messages will be received, regardless of filters and masks or the Receive Buffer Operating Mode bits, RXMn. The error counters are reset and deactivated in this state. The Listen-Only mode is activated by setting the Request Operation Mode bits (REQOP[2:0]) in the CANCTRL register.

2.1.5.4. Loopback Mode

Loopback mode will allow internal transmission of messages from the transmit buffers to the receive buffers without actually transmitting messages on the CAN bus. This mode can be used in system development and testing. The Loopback mode is activated by setting the Request Operation Mode bits in the CANCTRL register.

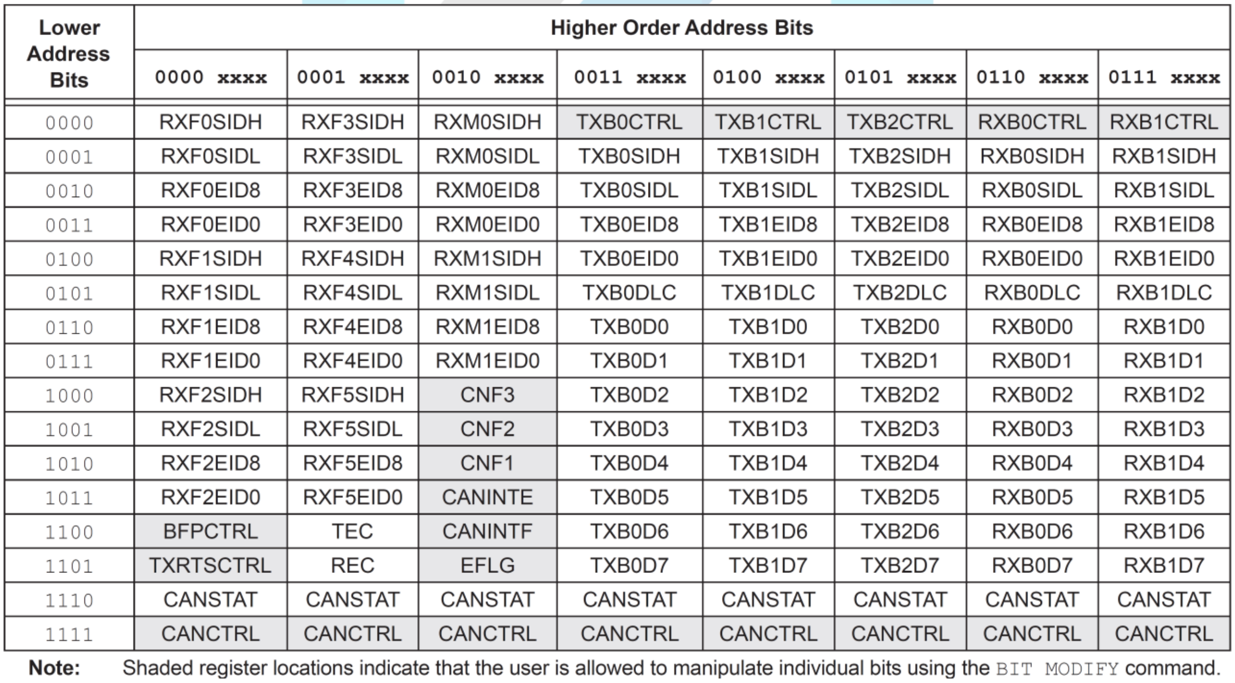
2.1.5.5. Normal Mode

Normal mode is the standard operating mode of the MCP2515. In this mode, the device actively monitors all bus messages and generates Acknowledge bits, error frames, etc. This is also the only mode in which the MCP2515 will transmit messages over the CAN bus.

2.1.6. Register Map

The register map for the MCP2515 is shown in Table 3-2. Address locations for each register are determined by using the column (higher order four bits) and row (lower order four bits) values. The registers have been arranged to optimize the sequential reading and writing of data. A summary of the MCP2515 control registers is shown in Table 3-3.

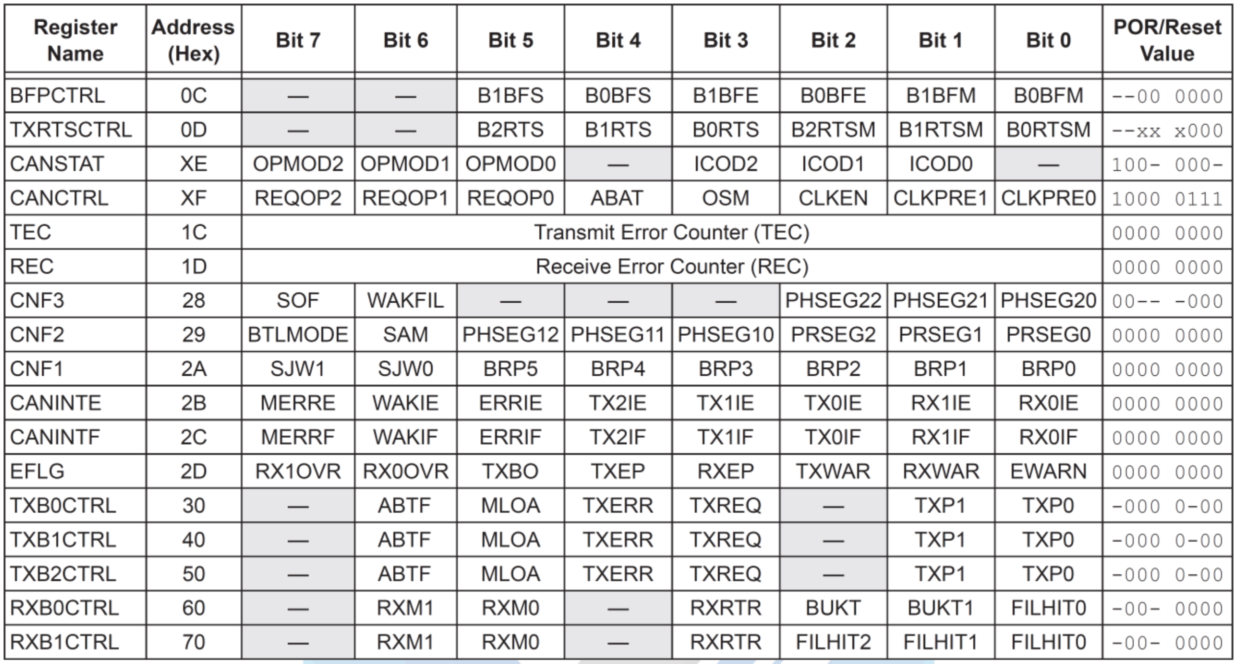
Table 3-2 CAN Controller Register Map



Lower Address Bits	Higher Order Address Bits							
	0000 xxxx	0001 xxxx	0010 xxxx	0011 xxxx	0100 xxxx	0101 xxxx	0110 xxxx	0111 xxxx
0000	RXF0SIDH	RXF3SIDH	RXM0SIDH	TXB0CTRL	TXB1CTRL	TXB2CTRL	RXB0CTRL	RXB1CTRL
0001	RXF0SIDL	RXF3SIDL	RXM0SIDL	TXB0SIDH	TXB1SIDH	TXB2SIDH	RXB0SIDH	RXB1SIDH
0010	RXF0EID8	RXF3EID8	RXM0EID8	TXB0SIDL	TXB1SIDL	TXB2SIDL	RXB0SIDL	RXB1SIDL
0011	RXF0EID0	RXF3EID0	RXM0EID0	TXB0EID8	TXB1EID8	TXB2EID8	RXB0EID8	RXB1EID8
0100	RXF1SIDH	RXF4SIDH	RXM1SIDH	TXB0EID0	TXB1EID0	TXB2EID0	RXB0EID0	RXB1EID0
0101	RXF1SIDL	RXF4SIDL	RXM1SIDL	TXB0DLC	TXB1DLC	TXB2DLC	RXB0DLC	RXB1DLC
0110	RXF1EID8	RXF4EID8	RXM1EID8	TXB0D0	TXB1D0	TXB2D0	RXB0D0	RXB1D0
0111	RXF1EID0	RXF4EID0	RXM1EID0	TXB0D1	TXB1D1	TXB2D1	RXB0D1	RXB1D1
1000	RXF2SIDH	RXF5SIDH	CNF3	TXB0D2	TXB1D2	TXB2D2	RXB0D2	RXB1D2
1001	RXF2SIDL	RXF5SIDL	CNF2	TXB0D3	TXB1D3	TXB2D3	RXB0D3	RXB1D3
1010	RXF2EID8	RXF5EID8	CNF1	TXB0D4	TXB1D4	TXB2D4	RXB0D4	RXB1D4
1011	RXF2EID0	RXF5EID0	CANINTE	TXB0D5	TXB1D5	TXB2D5	RXB0D5	RXB1D5
1100	BFPCTRL	TEC	CANINTF	TXB0D6	TXB1D6	TXB2D6	RXB0D6	RXB1D6
1101	TXRTSCTRL	REC	EFLG	TXB0D7	TXB1D7	TXB2D7	RXB0D7	RXB1D7
1110	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT	CANSTAT
1111	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL	CANCTRL

Note: Shaded register locations indicate that the user is allowed to manipulate individual bits using the BIT MODIFY command.

Table 3-3 Control Register Summary

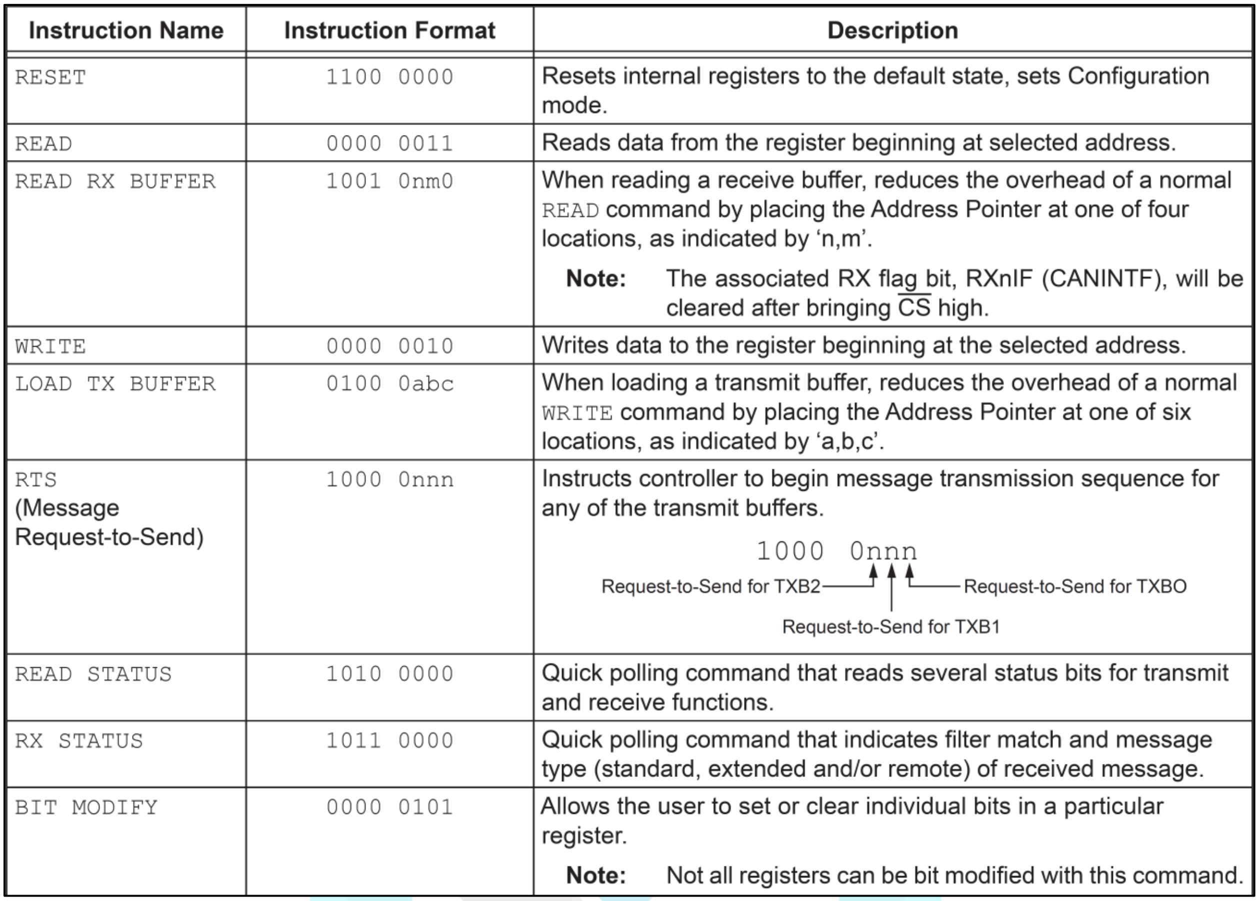


Register Name	Address (Hex)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR/Reset Value
BFPCTRL	0C	—	—	B1BFS	B0BFS	B1BFE	B0BFE	B1BFM	B0BFM	--00 0000
TXRTSCTRL	0D	—	—	B2RTS	B1RTS	B0RTS	B2RTSM	B1RTSM	B0RTSM	--xx x000
CANSTAT	XE	OPMOD2	OPMOD1	OPMOD0	—	ICOD2	ICOD1	ICOD0	—	100- 000-
CANCTRL	XF	REQOP2	REQOP1	REQOP0	ABAT	OSM	CLKEN	CLKPRE1	CLKPRE0	1000 0111
TEC	1C	Transmit Error Counter (TEC)								0000 0000
REC	1D	Receive Error Counter (REC)								0000 0000
CNF3	28	SOF	WAKFIL	—	—	—	PHSEG22	PHSEG21	PHSEG20	00-- -000
CNF2	29	BTLMODE	SAM	PHSEG12	PHSEG11	PHSEG10	PRSEG2	PRSEG1	PRSEG0	0000 0000
CNF1	2A	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	0000 0000
CANINTE	2B	MERRE	WAKIE	ERRIE	TX2IE	TX1IE	TX0IE	RX1IE	RX0IE	0000 0000
CANINTF	2C	MERRF	WAKIF	ERRIF	TX2IF	TX1IF	TX0IF	RX1IF	RX0IF	0000 0000
EFLG	2D	RX1OVR	RX0OVR	TXBO	TXEP	RXEP	TXWAR	RXWAR	EWARN	0000 0000
TXB0CTRL	30	—	ABTF	MLOA	TXERR	TXREQ	—	TXP1	TXP0	-000 0-00
TXB1CTRL	40	—	ABTF	MLOA	TXERR	TXREQ	—	TXP1	TXP0	-000 0-00
TXB2CTRL	50	—	ABTF	MLOA	TXERR	TXREQ	—	TXP1	TXP0	-000 0-00
RXB0CTRL	60	—	RXM1	RXM0	—	RXRTR	BUKT	BUKT1	FILHIT0	-00- 0000
RXB1CTRL	70	—	RXM1	RXM0	—	RXRTR	FILHIT2	FILHIT1	FILHIT0	-00- 0000

2.1.7. SPI Interface

The MCP2515 is designed to interface directly with the Serial Peripheral Interface (SPI) port available on many microcontrollers and supports Mode 0,0 and Mode 1,1. Commands and data are sent to the device via the SI pin, with data being clocked in on the rising edge of SCK. Data is driven out by the MCP2515 (on the SO line) on the falling edge of SCK. The CS pin must be held low while any operation is performed. Table 3-4 shows the instruction bytes for all operations. The MCP2515 expects the first byte after lowering CS to be the instruction/command byte. This implies that CS must be raised and then lowered again to invoke another command.

Table 3-4 SPI Instruction Set



Instruction Name	Instruction Format	Description
RESET	1100 0000	Resets internal registers to the default state, sets Configuration mode.
READ	0000 0011	Reads data from the register beginning at selected address.
READ RX BUFFER	1001 0nm0	When reading a receive buffer, reduces the overhead of a normal READ command by placing the Address Pointer at one of four locations, as indicated by ‘n,m’. **Note:** The associated RX flag bit, RXnIF (CANINTF), will be cleared after bringing $\overline{CS}$ high.
WRITE	0000 0010	Writes data to the register beginning at the selected address.
LOAD TX BUFFER	0100 0abc	When loading a transmit buffer, reduces the overhead of a normal WRITE command by placing the Address Pointer at one of six locations, as indicated by ‘a,b,c’.
RTS (Message Request-to-Send)	1000 0nnn	Instructs controller to begin message transmission sequence for any of the transmit buffers. 1000 0nnn Request-to-Send for TXB2 Request-to-Send for TXBO Request-to-Send for TXB1
READ STATUS	1010 0000	Quick polling command that reads several status bits for transmit and receive functions.
RX STATUS	1011 0000	Quick polling command that indicates filter match and message type (standard, extended and/or remote) of received message.
BIT MODIFY	0000 0101	Allows the user to set or clear individual bits in a particular register. **Note:** Not all registers can be bit modified with this command.

2.2. *MCP2551 CAN Transceiver*

The MCP2551 is a high-speed CAN, fault-tolerant device that serves as the interface between a CAN protocol controller and the physical bus. The MCP2551 provides differential transmit and receive capability for the CAN protocol controller. It will operate at speeds of up to 1 Mb/s. Typically, each node in a CAN system must have a device to convert the digital signals generated by a CAN controller to signals suitable for transmission over the bus cabling (differential output). It also provides a buffer between the CAN controller and the high-voltage spikes that can be generated on the CAN bus by outside sources (EMI, ESD, electrical transients, etc.).

Figure 3-1 shows the schematic diagram of how CAN Controller and CAN Transceiver are interfaced on board with the STM32G030F6P6 microcontroller.

2.2.1. *Features*

- Supports 1 Mb/s operation
- Implements ISO-11898 standard physical layer requirements
- Suitable for 12V and 24V systems
- Externally-controlled slope for reduced RFI emissions
- Detection of ground fault (permanent dominant) on TXD input
- Power-on reset and voltage brown-out protection
- An unpowered node or brown-out event will not disturb the CAN bus
- Low current standby operation
- Protection against damage due to short-circuit conditions (positive or negative battery voltage)
- Protection against high-voltage transients
- Automatic thermal shutdown protection
- Up to 112 nodes can be connected
- High noise immunity due to differential bus implementation
- Temperature ranges:
  - Industrial (I): -40°C to +85°C
  - Extended (E): -40°C to +125°C

2.2.2. *Pinout and Block Diagram*

Figure 3-5 shows the pin out diagram for MCP2551. An 8 – Lead PDIP/SOIC package of the IC is used on the board. Also the figure 3-6 shows the block diagram of CAN transceiver MCP2551. The 8-pin pinout description is listed in Table 3-5.

Figure 3-5 Pinout diagram of MCP2551

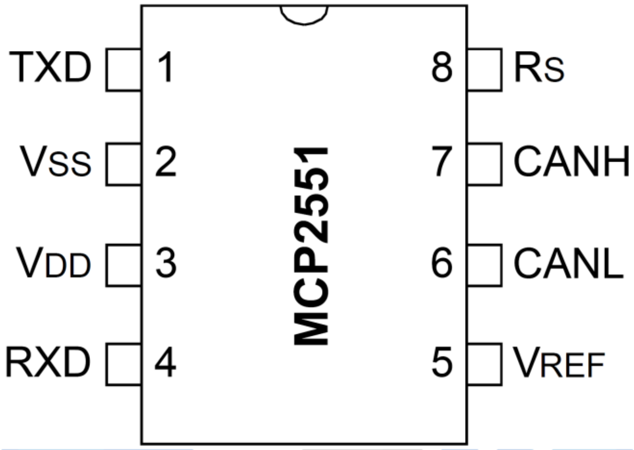
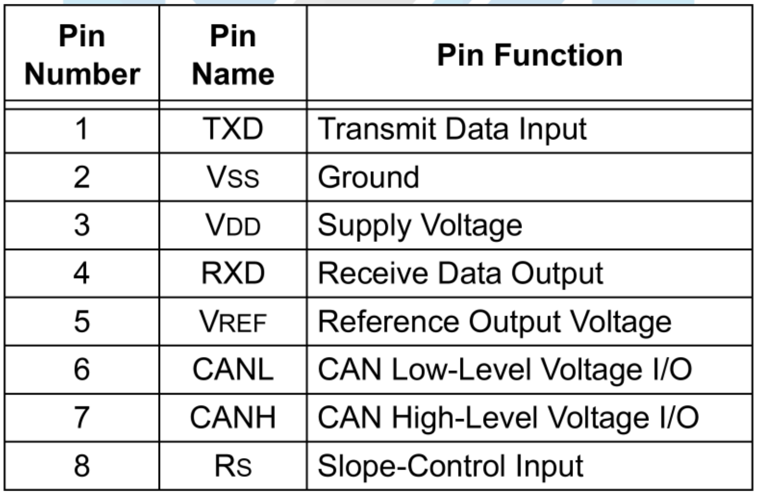
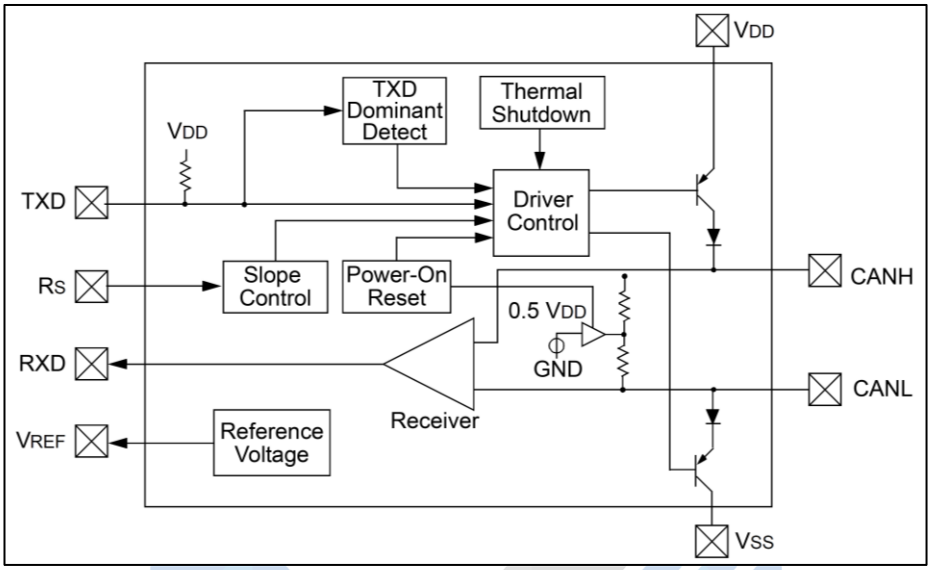


Table 3-5 MCP2551 Pinout Description



Pin Number	Pin Name	Pin Function
1	TXD	Transmit Data Input
2	VSS	Ground
3	VDD	Supply Voltage
4	RXD	Receive Data Output
5	VREF	Reference Output Voltage
6	CANL	CAN Low-Level Voltage I/O
7	CANH	CAN High-Level Voltage I/O
8	RS	Slope-Control Input

Figure 3-6 Block Diagram of MCP2551



2.2.3. Device Overview

2.2.3.1. Transmitter Function

The CAN bus has two states: Dominant and Recessive. A dominant state occurs when the differential voltage between CANH and CANL is greater than a defined voltage (e.g., 1.2V). A recessive state occurs when the differential voltage is less than a defined voltage (typically 0V). The dominant and recessive states correspond to the low and high state of the TXD input pin, respectively. However, a dominant state initiated by another CAN node will override a recessive state on the CAN bus.

MAXIMUM NUMBER OF NODES:

The MCP2551 CAN outputs will drive a minimum load of 45Ω, allowing a maximum of 112 nodes to be connected (given a minimum differential input resistance of 20 kΩ and a nominal termination resistor value of 120Ω).

2.2.3.2. Receiver Function

The RXD output pin reflects the differential bus voltage between CANH and CANL. The low and high states of the RXD output pin correspond to the dominant and recessive states of the CAN bus, respectively.

2.2.3.3. Internal Protection

CANH and CANL are protected against battery short circuits and electrical transients that can occur on the CAN bus. This feature prevents destruction of the transmitter output stage during such a fault condition. The device is further protected from excessive current loading by thermal shutdown circuitry that disables the output drivers when the junction temperature exceeds a nominal limit of 165°C.

2.2.3.4. Operating Modes

The RS pin allows three modes of operation to be selected:

- High-Speed
- Slope-Control
- Standby

When in High-speed or Slope-control mode, the drivers for the CANH and CANL signals are internally regulated to provide controlled symmetry in order to minimize
EMI emissions. Additionally, the slope of the signal transitions on CANH and CANL can be controlled with a resistor connected from pin 8 (RS) to ground, with the slope proportional to the current output at RS, further reducing EMI emissions.

HIGH-SPEED:

High-speed mode is selected by connecting the RS pin to VSS. In this mode, the transmitter output drivers have fast output rise and fall times to support high-speed
CAN bus rates.

SLOPE-CONTROL:

Slope-control mode further reduces EMI by limiting the rise and fall times of CANH and CANL. The slope, or slew rate (SR), is controlled by connecting an external resistor (REXT) between RS and VOL (usually ground). The slope is proportional to the current output at the RS pin. Since the current is primarily determined by the slope-control resistance value REXT, a certain slew rate is achieved by applying a respective resistance.

STANDBY MODE:

The device may be placed in standby or “SLEEP” mode by applying a high-level to RS. In SLEEP mode, the transmitter is switched off and the receiver operates at a lower current. The receive pin on the controller side (RXD) is still functional but will operate at a slower rate. The attached microcontroller can monitor RXD for CAN bus activity and place the transceiver into normal operation via the RS pin (at higher bus rates, the first CAN message may be lost).

Appendix A

Pinout and PCB Layout

Figure A-1 STM32G030Fx TSSOP20 pinout

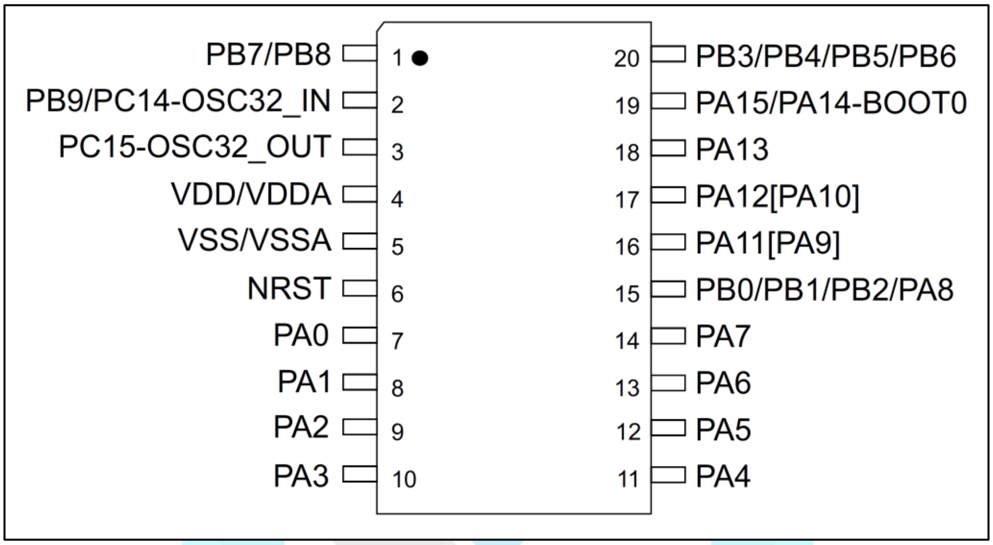
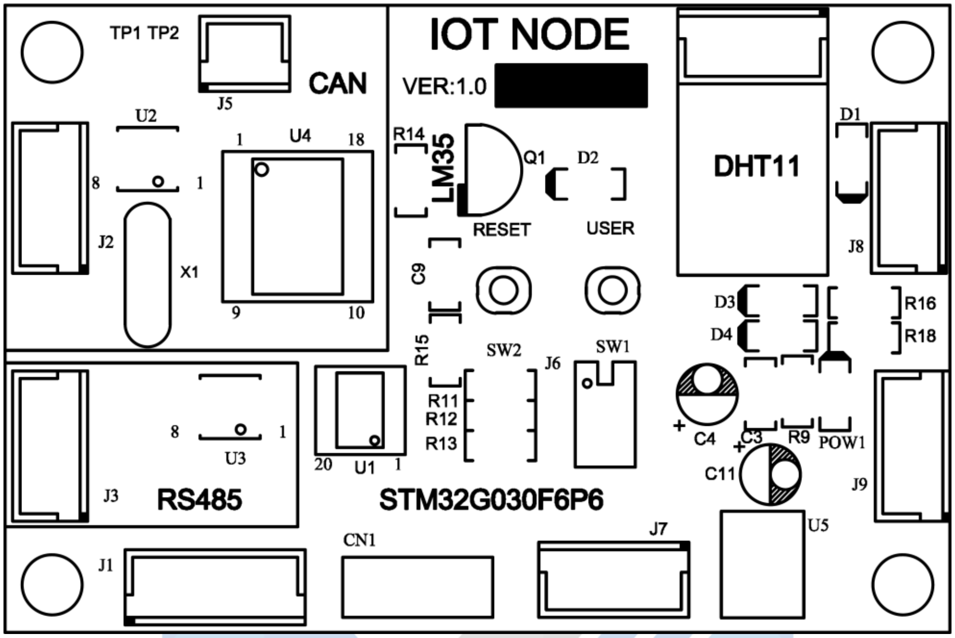
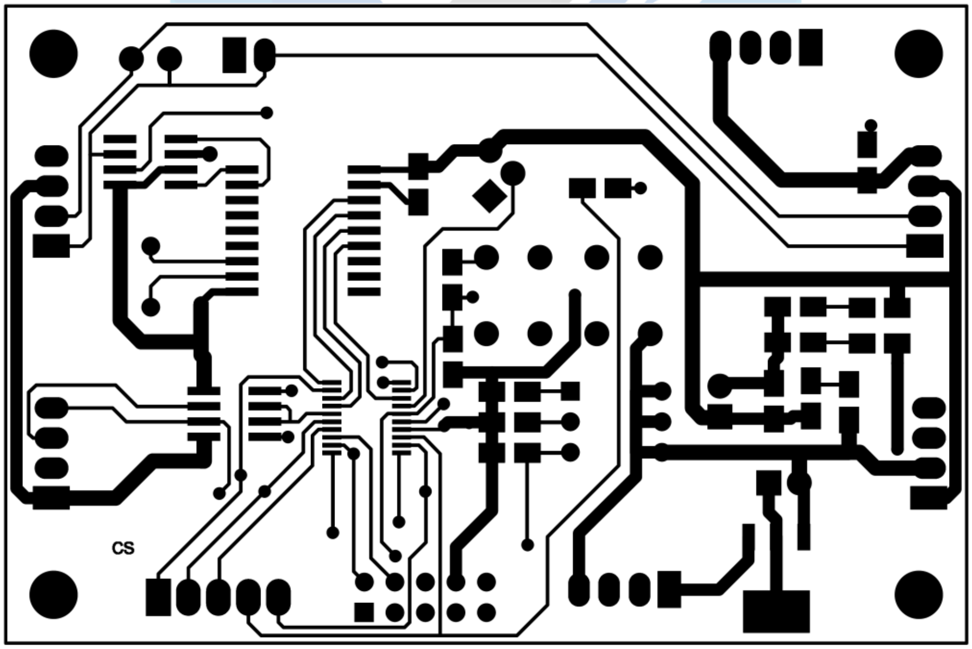
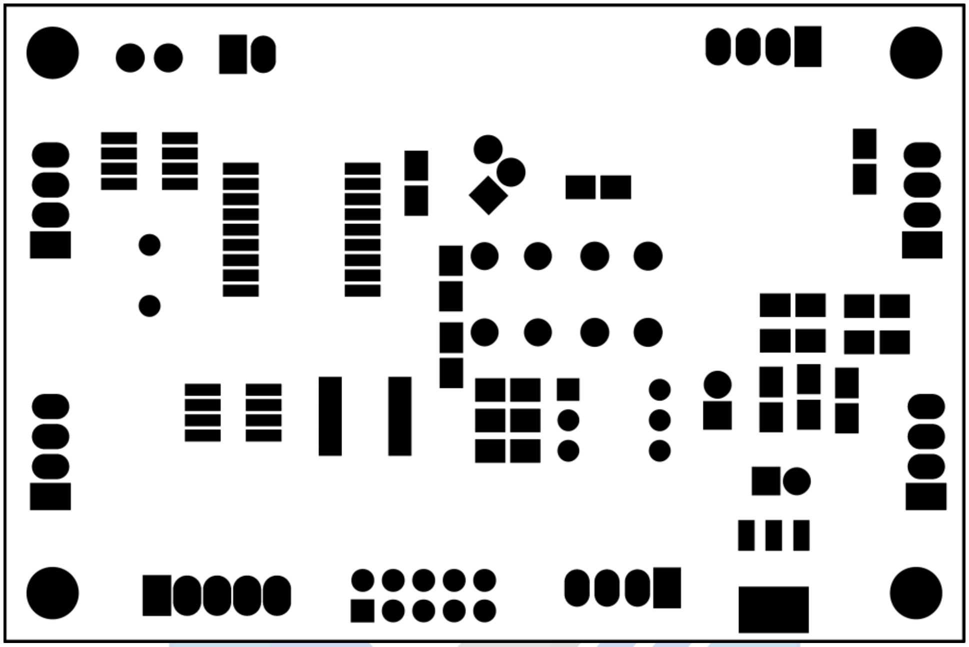
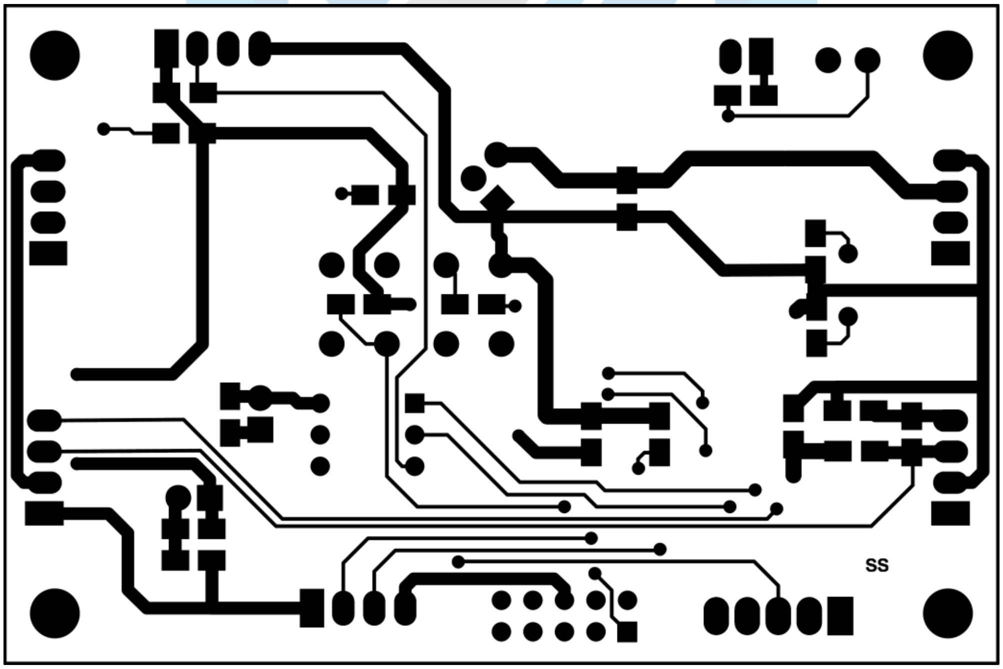


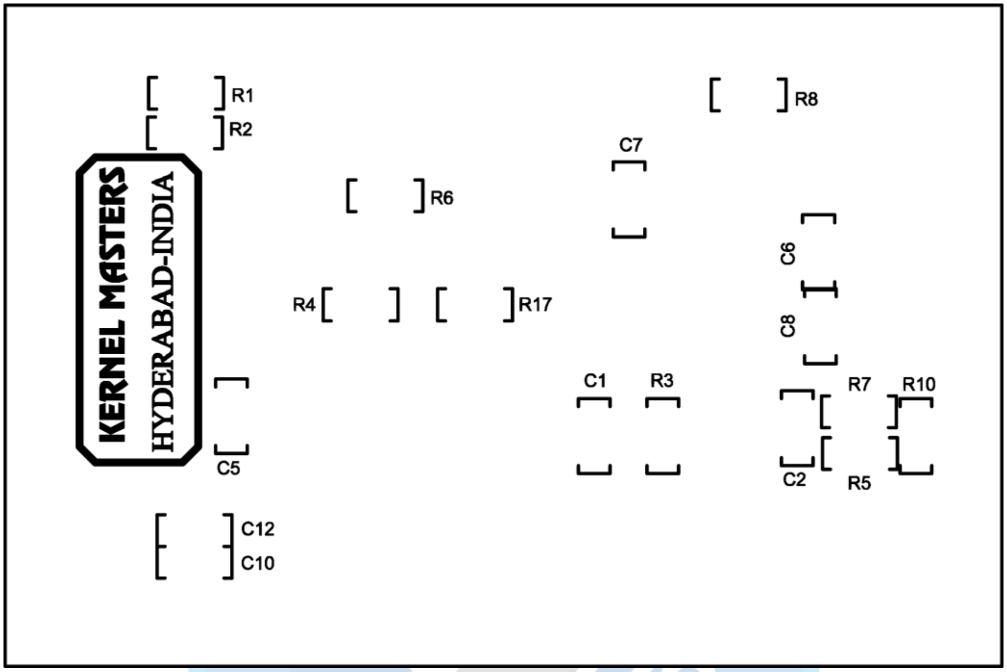
Figure A-2 IoT Node Board PCB Layout


TP1 TP2
J5
CAN
IOT NODE
VER:1.0
DHT11
D1
U2
1
U4
18
R14
LM35
Q1
D2
8
1
J2
X1
RESET
USER
J8
C9
9
10
D3
D4
R16
R18
R15
SW2
SW1
J6
R11
R12
R13
C4
C3
R9
POW1
C11
8
1
U3
20
U1
1
J3
RS485
STM32G030F6P6
J9
U5
J1
CN1
J7


CS



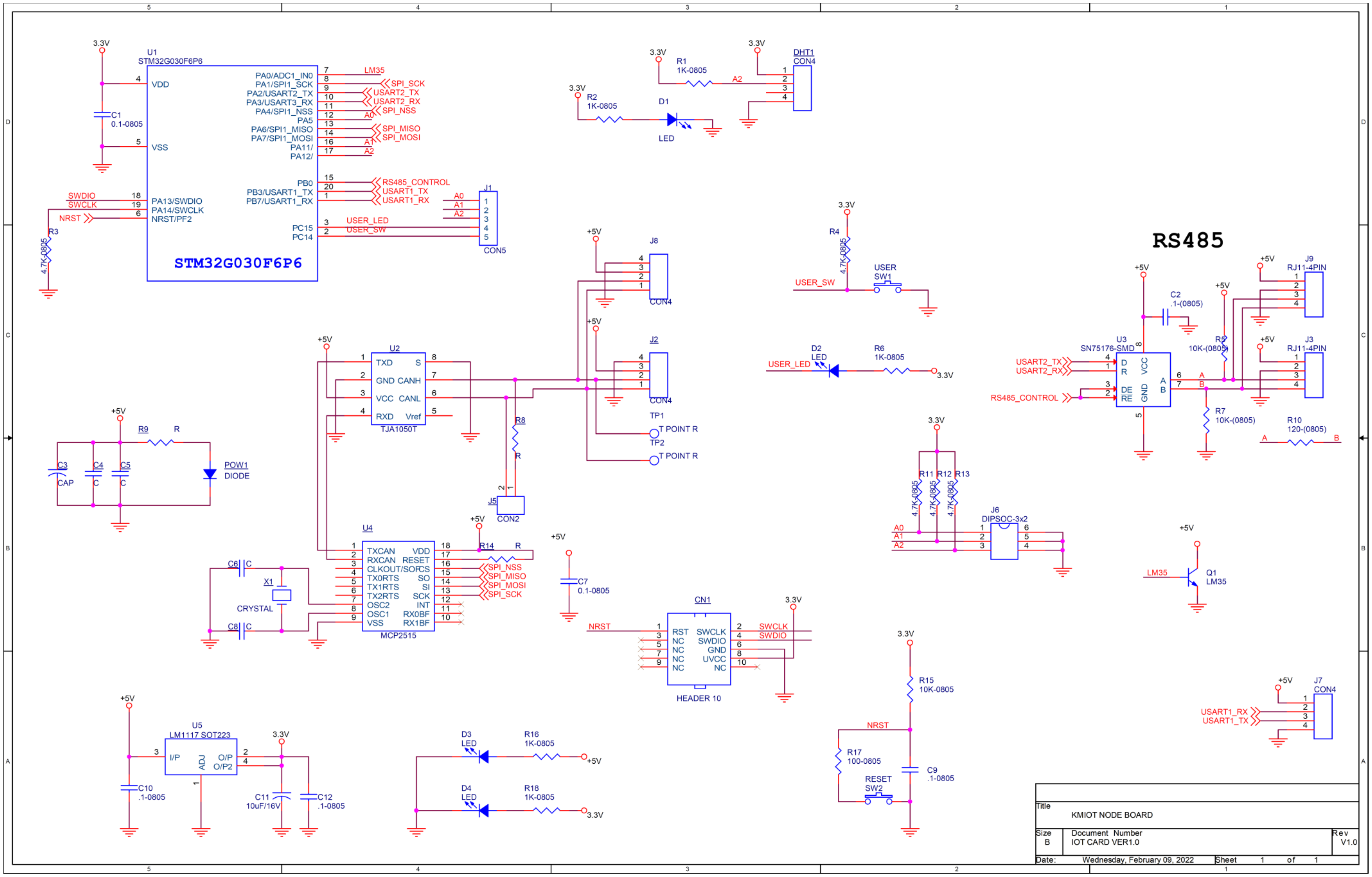

SS


R1
R2
R8
C7
R6
KERNEL MASTERS
HYDERABAD-INDIA
C6
R4
R17
C8
C1
R3
R7
R10
C5
C2
R5
C12
C10

Appendix B

SCHEMATICS

This section contains the complete schematics for the CAN IoT Node Development board.


U1
STM32G030F6P6
VDD
VSS
PA0/ADC1_IN0
PA1/SPI1_SCK
PA2/USART2_TX
PA3/USART3_RX
PA4/SPI1_NSS
PA5
PA6/SPI1_MISO
PA7/SPI1_MOSI
PA11/
PA12/
PB0
PB3/USART1_TX
PB7/USART1_RX
PA13/SWDIO
PA14/SWCLK
NRST/PF2
PC15
PC14
STM32G030F6P6
LM35
SPI_SCK
USART2_TX
USART2_RX
SPI_NSS
SPI_MISO
SPI_MOSI
A0
A1
A2
RS485_CONTROL
USART1_TX
USART1_RX
USER_LED
USER_SW
SWDIO
SWCLK
NRST
C1
0.1-0805
R3
4.7K-0805
J1
CON5
3.3V
+5V
R1
1K-0805
R2
1K-0805
D1
LED
DHT1
CON4
J8
CON4
J2
CON4
TP1
T POINT R
TP2
T POINT R
U2
TXD
S
GND CANH
VCC CANL
RXD Vref
TJA1050T
R8
R
J5
CON2
R4
4.7K-0805
USER
SW1
D2
LED
R6
1K-0805
RS485
U3
SN75176-SMD
D
R
DE
RE
VCC
GND
A
B
C2
.1-(0805)
R5
10K-(0805)
R7
10K-(0805)
R10
120-(0805)
J9
RJ11-4PIN
J3
RJ11-4PIN
R9
R
C3
CAP
C4
C
C5
C
POW1
DIODE
U4
TXCAN VDD
RXCAN RESET
CLKOUT/SOFCS
TX0RTS SO
TX1RTS SI
TX2RTS SCK
OSC2 INT
OSC1 RX0BF
VSS RX1BF
MCP2515
R14
R
C6
C
C8
C
X1
CRYSTAL
C7
0.1-0805
R11
R12
R13
4.7K-0805
J6
DIPSOC-3x2
Q1
LM35
CN1
RST SWCLK
NC SWDIO
NC GND
NC UVCC
NC NC
HEADER 10
U5
LM1117 SOT223
I/P
ADJ
O/P
O/P2
C10
.1-0805
C11
10uF/16V
C12
.1-0805
D3
LED
D4
LED
R16
1K-0805
R18
1K-0805
R15
10K-0805
R17
100-0805
RESET
SW2
C9
.1-0805
J7
CON4
Title
KMIOT NODE BOARD
Size
B
Document Number
IOT CARD VER1.0
Rev
V1.0
Date: Wednesday, February 09, 2022
Sheet 1 of 1