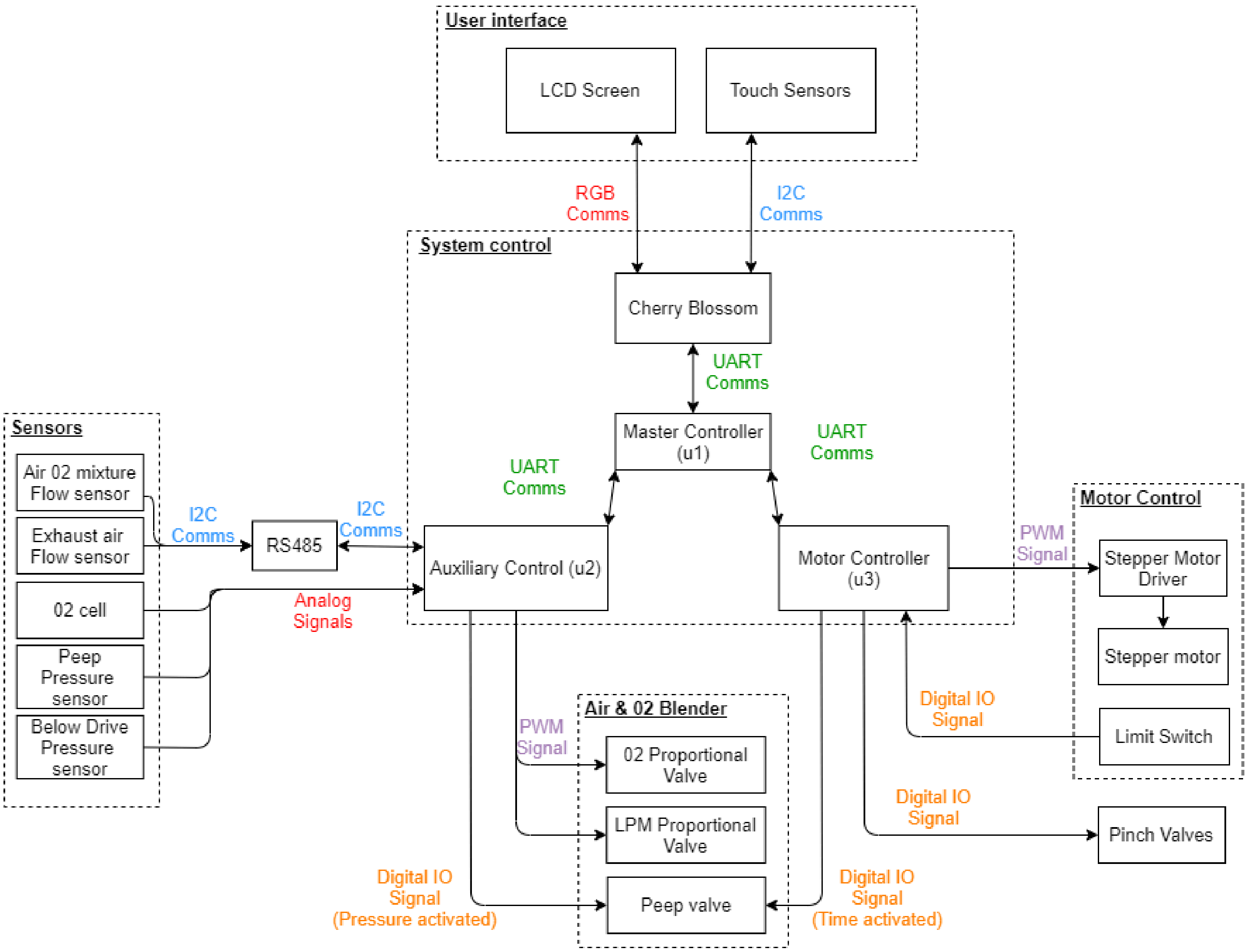
Ventilator Research and prototype

Nicholas Antoniades

2019

Table of Contents

1. System Overview



2. System functions

1. Cherry Blossom

1. Pushes functional states (setpoints) to u1 (Clinician is physically encoding their settings).
2. Receives sensor values and state feedback from u1.
3. Locally storing event log (any action).

2. Master controller (u1)

1. Receives functional states from Cherry.
2. Updates functional state on u2 (%02, 02 LPM, alarm setpoints).
3. Updates functional state on u3 (tidal volume, bpm, inspiration time).
4. Decide ramp rates for modifying setpoints.
5. u1 pulls sensor values from u2 and pushes them to the Cherry.
6. u1 pulls state feedback from u2 and u3 and push=es them to the Cherry.

3. Slave Motor controller (u2)

1. Receive state update from u1.
2. Pushes current state to u1.
3. Functional control of the stepper motor.
4. Calibration of the stepper motor using the limit switch.
5. Functional control of pinch valves.
6. Functional control of PEEP solenoid. Valve is opened based on the timing of the inhalation cycle.

4. Slave Auxiliary controller (u3)

1. Receive alarm functional states from u1.
2. Pushes sensor data and current state to u1.
3. Functional control by PEEP solenoid.
4. Valve is closed through digital IO output from u2 based on pressure signal.

5. Alarm interrupts are monitored and can initiate the following actions:

1. No action.
2. Feedback protection (Push functional state to u1)
3. User feedback (Receive new functional state from u1)

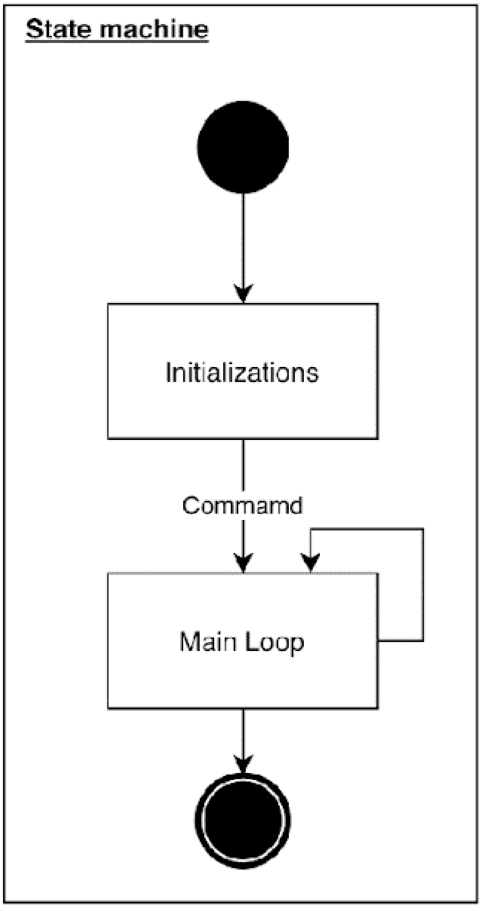
3. Inputs and outputs

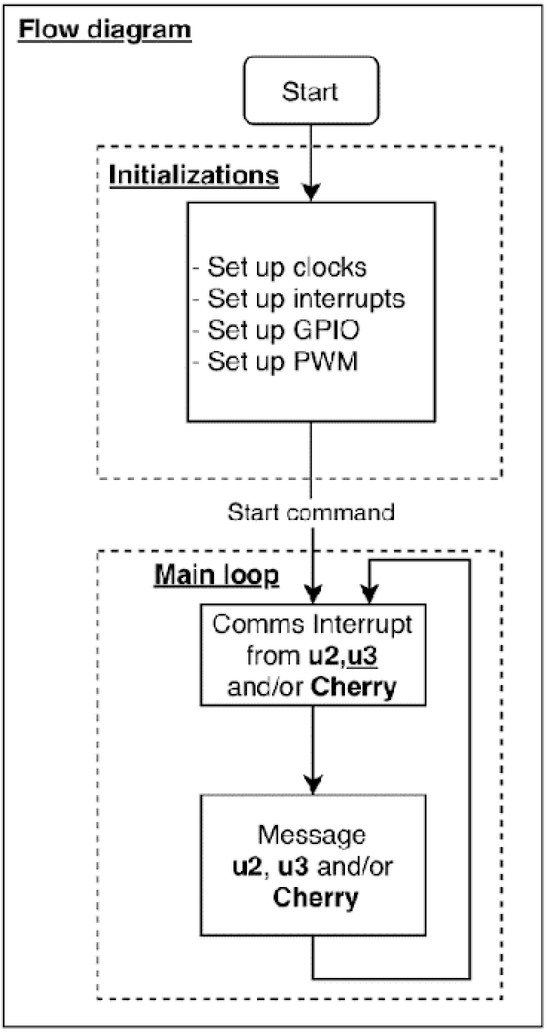
TABLE 1: ACTUATION		
NO.	Operation control	Feedback protection
1	Stepper motor	PIP exceeded (Solenoid)
2	PEEP Solenoid	
3	Pinch Valve Inlet (Solenoid)	
4	Pinch Valve Outlet (Solenoid)	
5	Oxygen % Solenoid	
6	Oxygen LPM Solenoid	

TABLE 2: INPUTS AND FEEDBACK		
NO.	Sensors	User feedback (alarms) (Both fixed and user specific)
1	Mass flow rate sensor (i2c)	Power supply failure
2	Pressure sensor (Analog)	Gas supply failure
3	Fi02 sensor (Analog)	Tidal volume not achieved
4	C02 sensor (Analog)	PIP not achieved
5	Oxygen Saturation Temperature (Analog)??	BPM not achieved
6	Tidal Volume (Soft sensor)	PEEP not achieved
7	Limit switch (digital)	

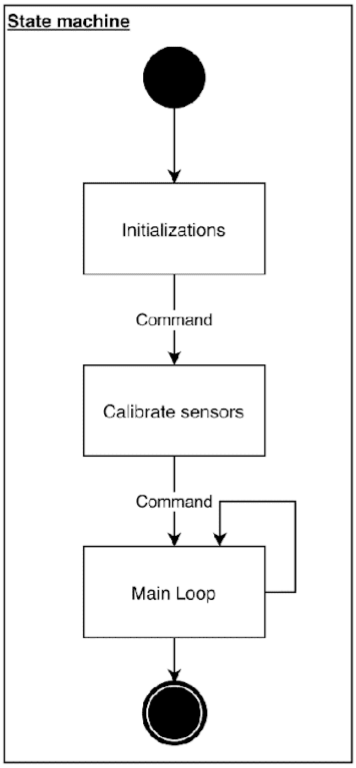
4. State machines and flow diagrams

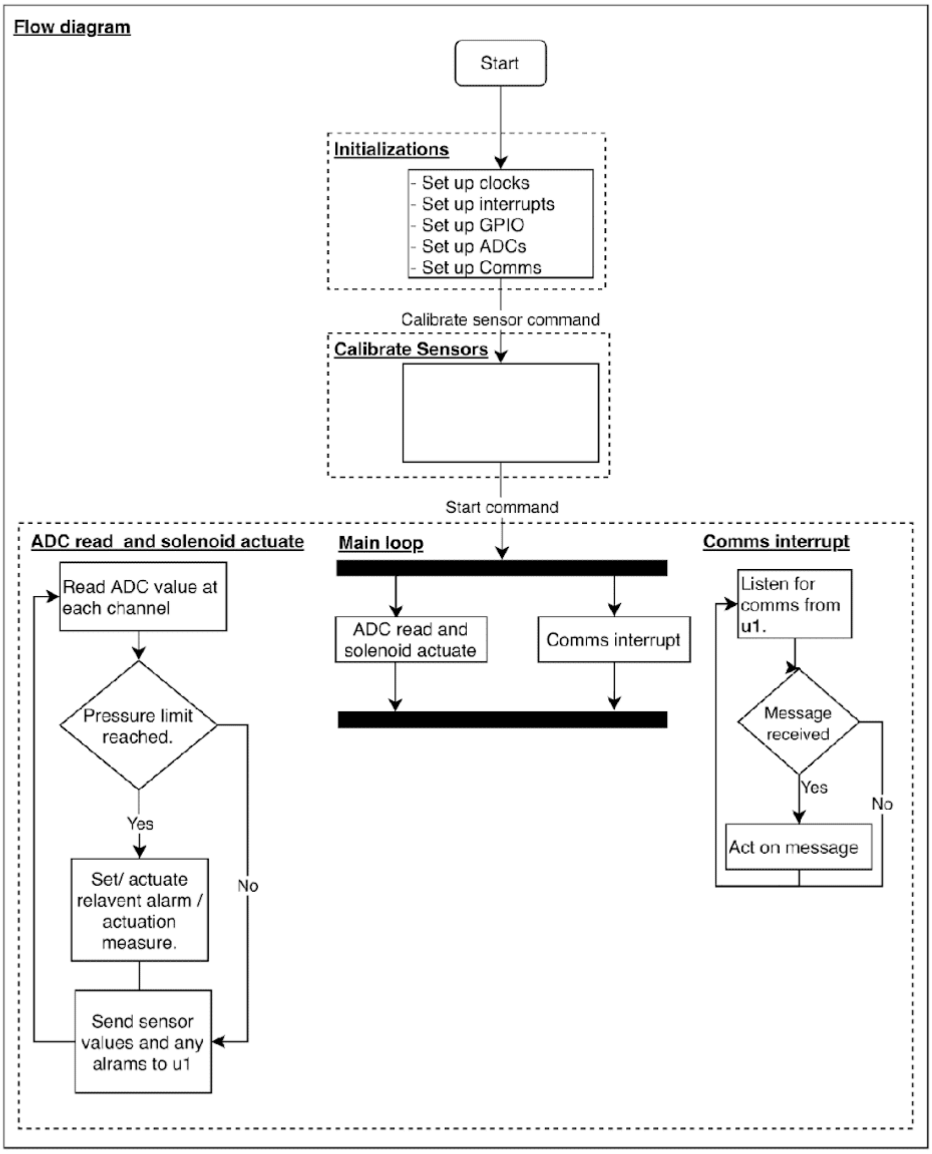
1. Master controller (u1)



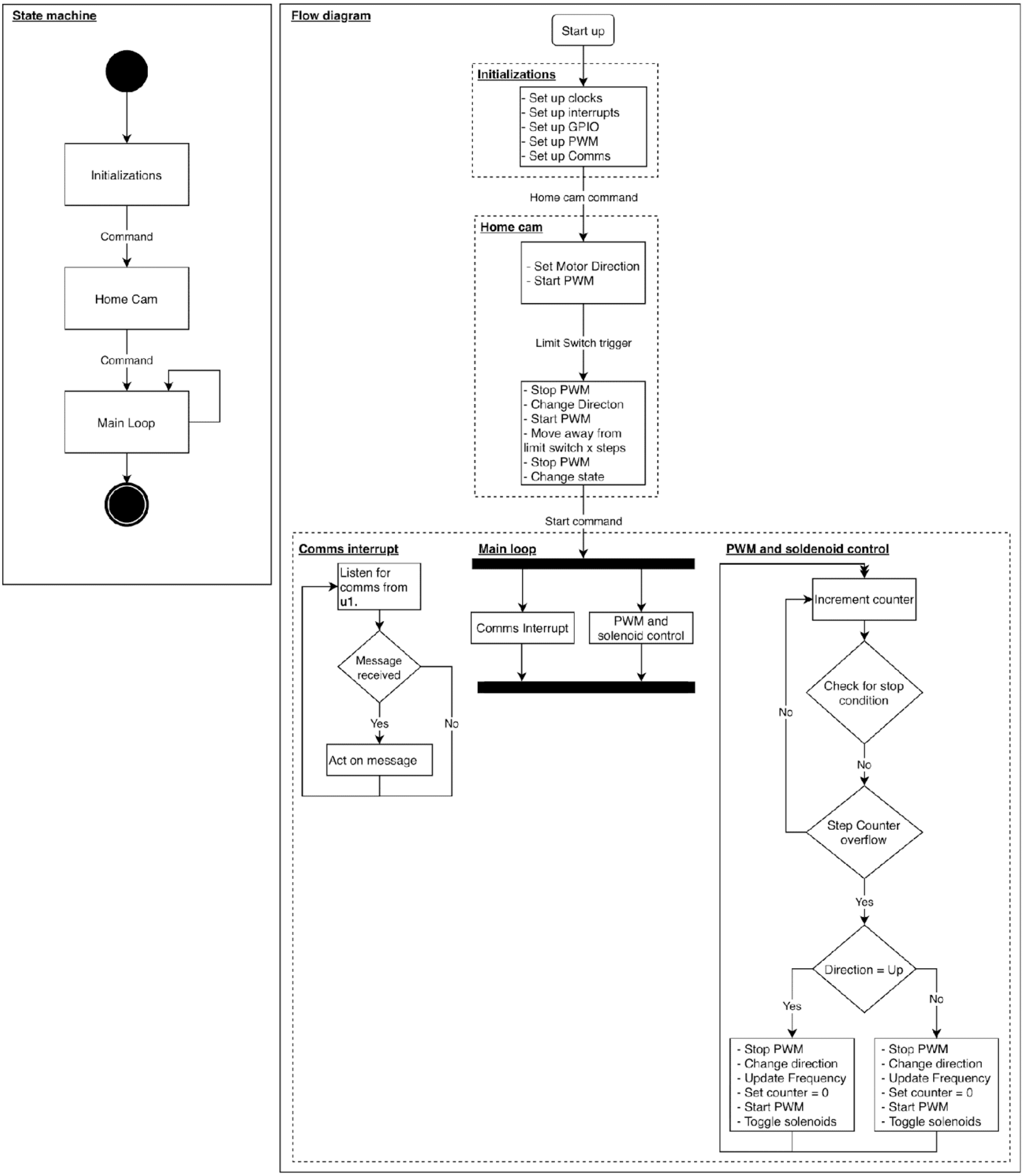


2. Auxiliary control (u2)





3. Motor Controller (u3)

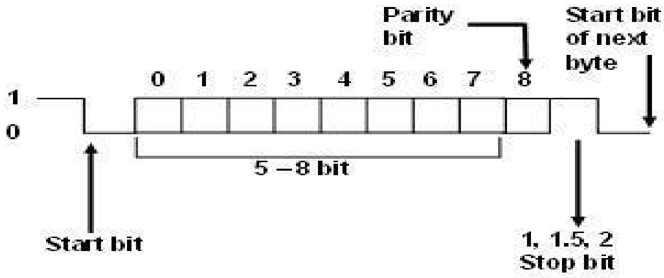


5. Relevant Jargon

TABLE 3: EXAMPLE OF ANOTHER TABLE HEADER		
NO.	Word	Definition
1	VentMethod	Numerical indication of the required ventilation method.
2	stepCounter	Counter used to track each step of the motor.
3	cycleStage	Indicates whether system is in Inspiration or expiration half of cycle. cycleStage = 1 for Expiration.
4	gearRatio	Gear ratio determined by rack and gear driven by the motor.
5	microStep	Set on the stepper motor driver, effects total number of per revolution of the motor.
6	StepsLimitSwitch	The number of steps to move away from the limit switch after motor has homed.
7	TotalStepsPerCycle	The total number of steps for a full inspiration and expiration cycle
8	StepsHalfCycle	Half of the total number of steps for a full inspiration and expiration cycle.
9	RequiredStepsHalfCycle	The required steps based on StepsHalfCycle and desired volume to be delivered.
10	uartBufferSize	Size of the buffer required for the UART communications
11	BPM	Beats Per Minute.
12	IEratio	Inspiration to Expiration ratio.
13	Ttotal	Total cycle time.
14	InspTime	Inspiration time.
15	ExpTime	Expiration time.
16	InspFreq	Inspiration Frequency.
17	ExpFreq	Expiration Frequency.
18	ExpPSC	Prescalar value for the required PWM Expiration frequency.
19	InspPSC	Prescalar value for the required PWM Inspiration frequency.
20	Vmax	Max deliverable volume.
21	Vdes	Desired volume to deliver.

6. UART Message Structure

4. Message structure



Each byte of data being sent will be sent between a start, stop and a parity bit. Each message being sent will be made up from multiple bytes. The first byte, byte 0. Will represent the type of message being sent. The following bytes, bytes 1 to n, will contain any other required information of the message. Each byte that is received will be added to a buffer in the order they arrived.

Buffer example:

byte 0	message type
byte 1	message information
|	
byte n	message information

Table 4: Cherry blossom nd mastear

NO.	Cherry -> Master(u1)	Master(u1)->Cherry
1	byte 0 = 1, Initialize system	Power supply failure
2	byte 0 = 2, Calibrate system	Gas supply failure
3	byte 0 = 3, Request current system state	Tidal volume not achieved
4	byte 0 = 4, Stop and reset command	BPM not achieved
5	byte 0 = 5, Update Ventilation byte 1 = Ventilation Method byte 2 = BPM byte 3 = Tidal Volume byte 4 = IEratio byte 5 = Alarm set points	PEEP not achieved

TABLE 5: MASTER AND MOTOR CONTROL		
NO.	Master(u1)->Motor Control(u2)	Motor Control(u2)->Master(u1)
1	byte 0 = 1. Initialize system	byte 0 = 1. Current state
2	byte 0 = 2. Calibrate system	byte 0 = 2. Alarm states
3	byte 0 = 3. Request current system state	byte 0 = 3. Any errors that may occur
4	byte 0 = 4. Stop and reset command	
5	byte 0 = 5, Update Ventilation byte 1 = Ventilation Method byte 2 = BPM byte 3 = Tidal Volume byte 4 = IEratio byte 5 = Alarm set points	

TABLE 6: MASTER AND AUXILIARY CONTROL		
NO.	Master(u1)->Motor Control(u2)	Motor Control(u2)->Master(u1)
1	byte 0 = 1. Current state	byte 0 = 1. Current state
2	byte 0 = 2. Update Sensor set points	byte 0 = 2. Sensor values
3	byte 0 = 3. Alarm states	byte 0 = 3. Alarm states
4		byte 0 = 4. Any errors that may occur

7. Calculations

Solving for required PWM frequency based on Ventilator settings:

1. Ttotal = 60/BPM

2. IEratio = InspTime/ExpTime
   InspTime = IEratio*ExpTime

3. Ttotal = InspTime + ExpTime
   Ttotal = IEratio*ExpTime + ExpTime
   ExpTime = Ttotal(1/(1+IEratio))
   InspTime = Ttotal(IEratio/(1 + IEratio))

4. TotalStepsPerCycle = TotalStepsPerCycle*microStep*gearRatio
   StepsHalfCycle = TotalStepsPerCycle/2
   RequiredStepsHalfCycle = (StepsHalfCycle*Vdes)/(Vmax)

   InspFreq = RequiredStepsHalfCycle/InspTime
   ExpFreq = RequiredStepsHalfCycle/ExpTime

5. ExpPSC = ((float)clockFreq/counterPeriod)/ExpFreq
   InspPSC = ((float)clockFreq/counterPeriod)/InspFreq

8. Final PCB

a. U1, U2 and U3 peripheral circuitry

- Voltage regulator
- For decreased power consumption set all unused pins to analog mode and connect them to ground.
- Filters for:
    - Communication channels
    - ADC channels,
    - Power and Ground.
    - Digital IOs
- Possible buffers or amplifiers for sensors.
- Connectors between controller board and sensors/ actuators.
- Testing pads on the board for debugging during assembly.
- Connectors for the debugger.
- Correct crystal oscillator.
- Extra connectors in case more peripherals are used.
- Possibly RS232 or RS485 chips for i2c comms with sensors.
- Digital IO and comms protection
    - Opto-couplers
    - TVS diodes to clamp voltage

b. Questions

1. How many ventilation types
2. Which sensors are being calibrated? How do they perform calibration live, are they adjusting drift or zero-ing channels?
3. How to connect the micros to update software via the interface device.

9. Annexure A: STM32F437VI

The STM32F437xx devices are based on the high-performance Arm Cortex-M4 32-bit RISC core operating at a frequency of up to 180 MHz. The Cortex-M4 core features a Floating-point unit (FPU) single precision. It also implements a full set of DSP instructions and a memory protection unit (MPU) which enhances application security.

The STM32F437xx devices incorporate high-speed embedded memories (Flash memory up to 2 Mbyte, up to 256 Kbytes of SRAM), up to 4 Kbytes of backup SRAM, and an extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB buses and a 32-bit multi-AHB bus matrix.

All devices offer three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers, a true random number generator (RNG) and a cryptographic acceleration cell. They also feature standard and advanced communication interfaces.

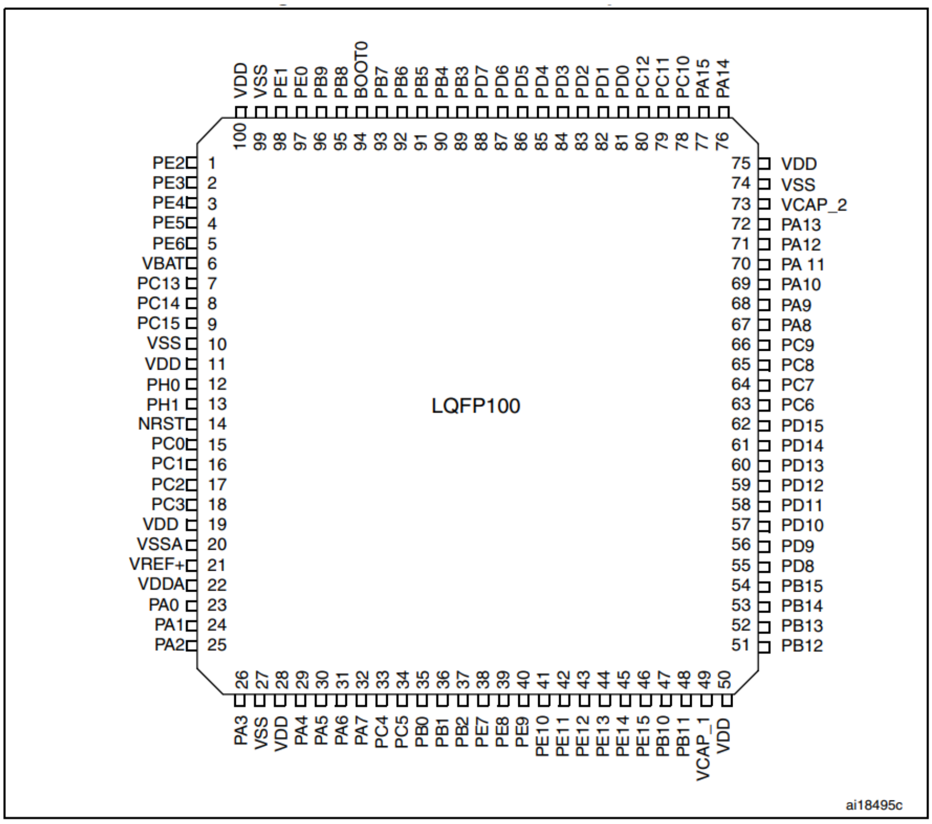


Figure : STM32F437xx LQFP100 package

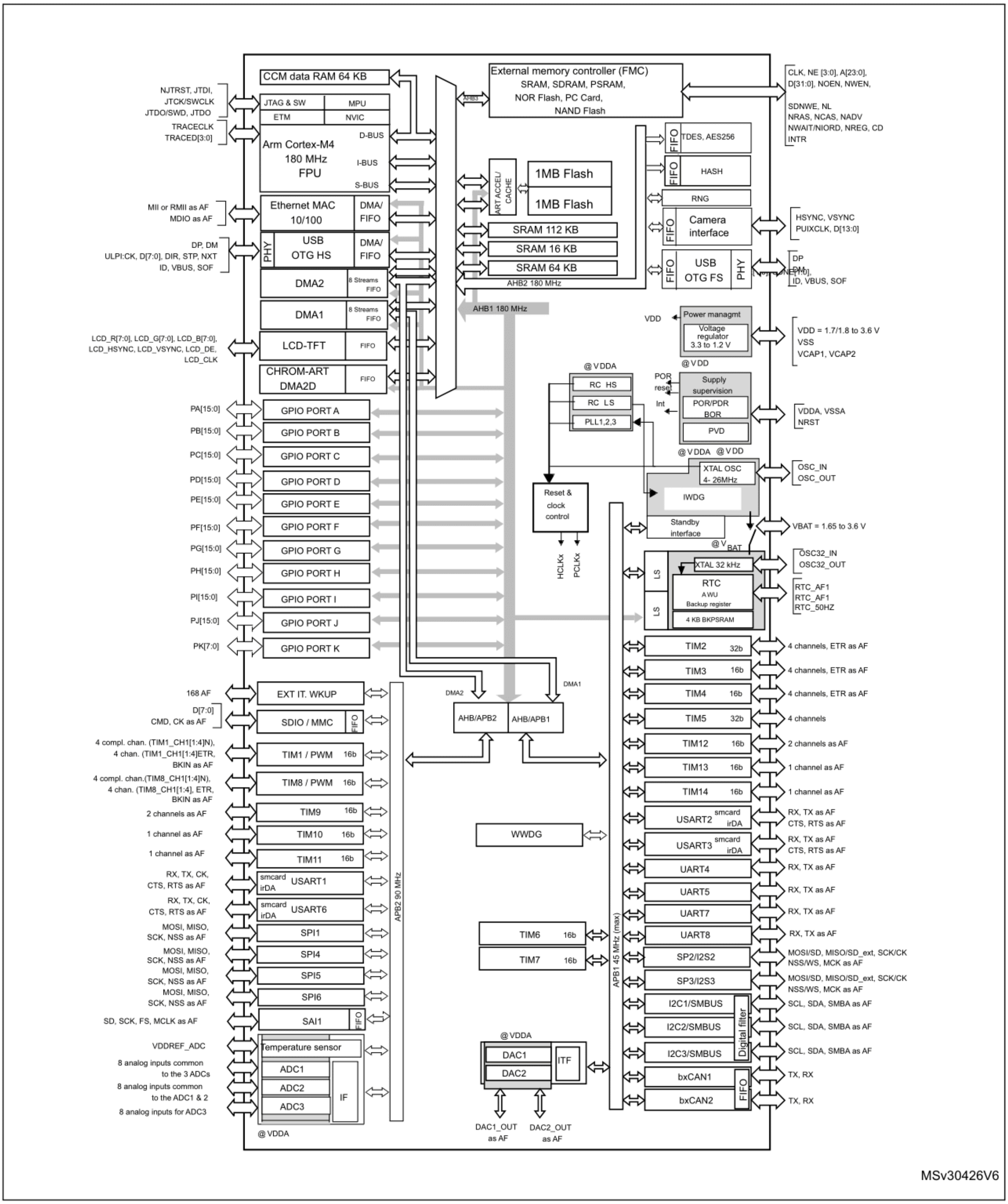


Figure : STM32F437xx Block Diagram

Coretex-M4 Processor

The Cortex-M4 processor is a low-power processor that features low gate count, low interrupt latency, and low-cost debugging. The Cortex-M4 includes optional floating-point arithmetic functionality. The processor is intended for deeply embedded applications that require fast interrupt response features.

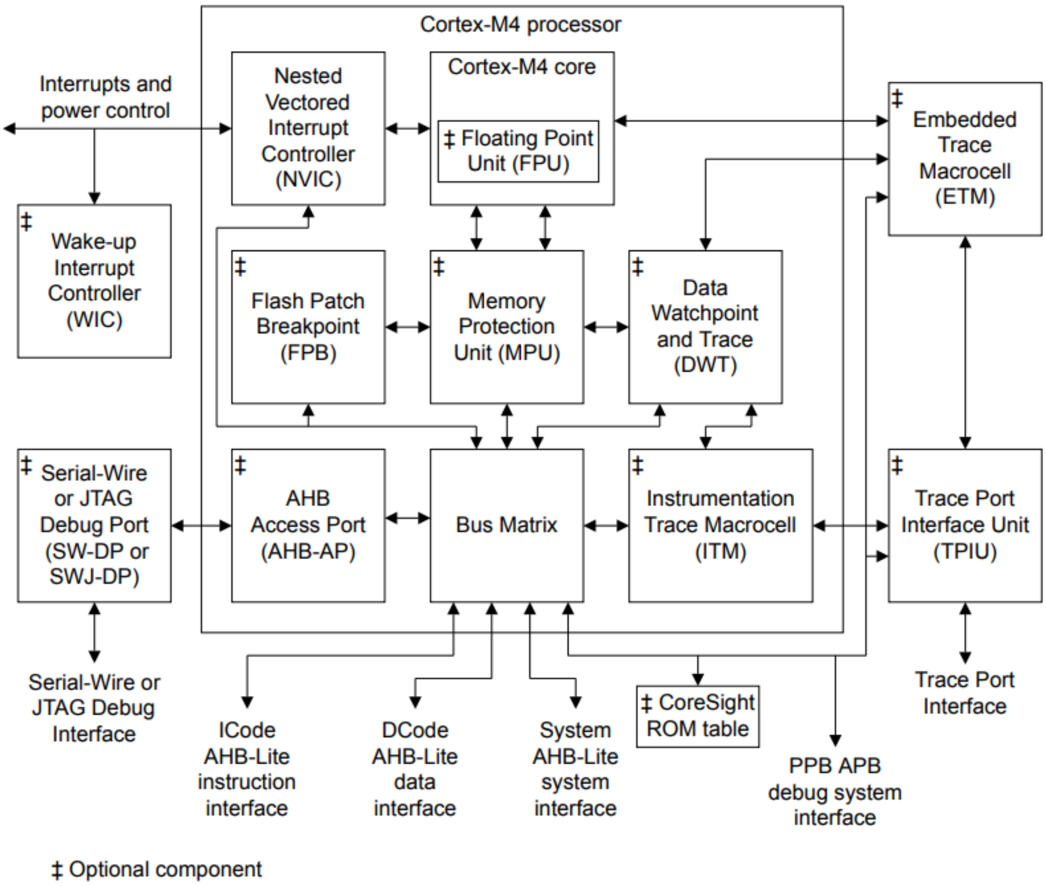


Figure : Arm-Cortex M4 processor Block Diagram

10. Annexure B: Communication types

UART

Universal Asynchronous Reception and Transmission (UART), is a communication where tow devices communicate directly with each other. TX and RX are connected between two devices

UARTs transmit data asynchronously, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead of a clock signal, the transmitting UART adds start and stop bits to the data packet being transferred. These bits define the beginning and end of the data packet so the receiving UART knows when to start reading the bits.

Pros:
- No clock needed
- parity bit to allow for error checking
- Two wires

Cons:
- Size of data frame is limited to only 9 bits
- Cannot use multiple master systems and slaves

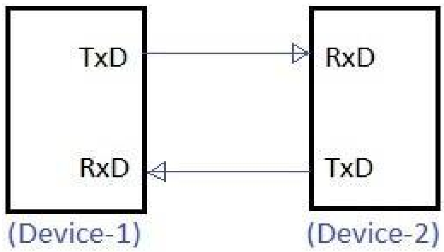
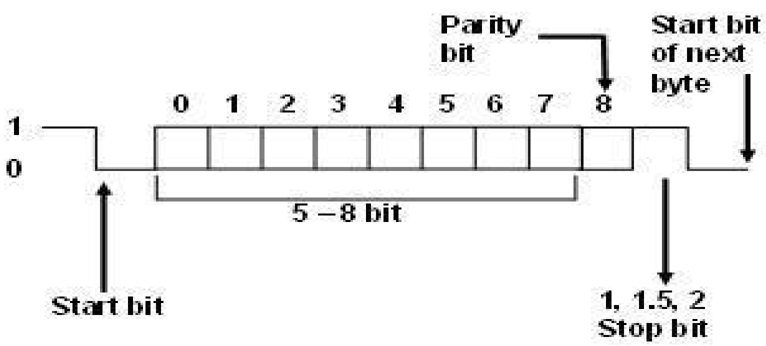


Figure : UART interface diagram



I2C

Inter-integrated-circuit (I2C). I2C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems. I2C bus is popular because it is simple to use, there can be more than one master. Each slave device has a unique address. SCL and SDA are connected between a master and multiple slaves. SCL synchronises the transmission of the data over SDA

Pros:

- Supports multi master and multi slave communication
- Two wires
- Can be faster than UART

Cons:

- May become complex as the number of devices increases.
- Message structure is more complicated that UART

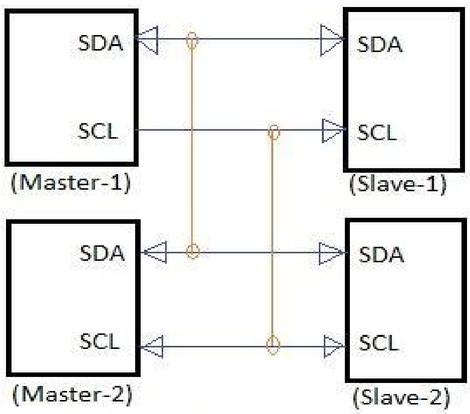
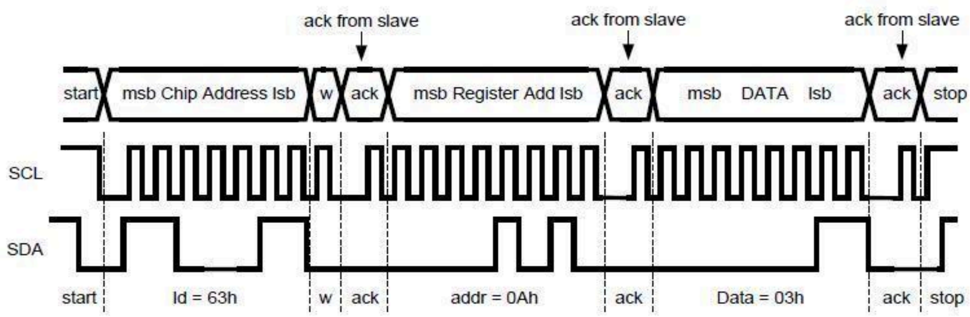


Figure : I2C interface diagram



RS485

RS485 is a standard defining the electrical characteristics of drivers and receivers for use in serial communications systems. RS485 allows for communications over distances of up to 1200 meters using differential signalling over a twisted pair cable. It can be used with data rate of up to 10Mbit/s for up to 50m and 2 Mbit/s at longer distances.

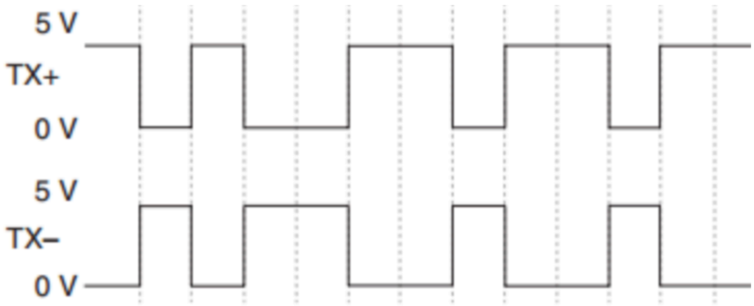
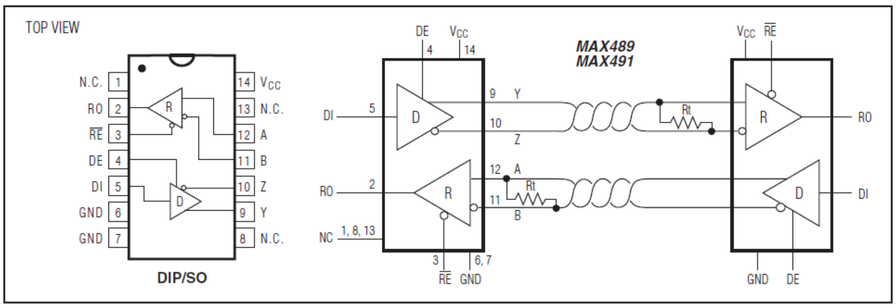
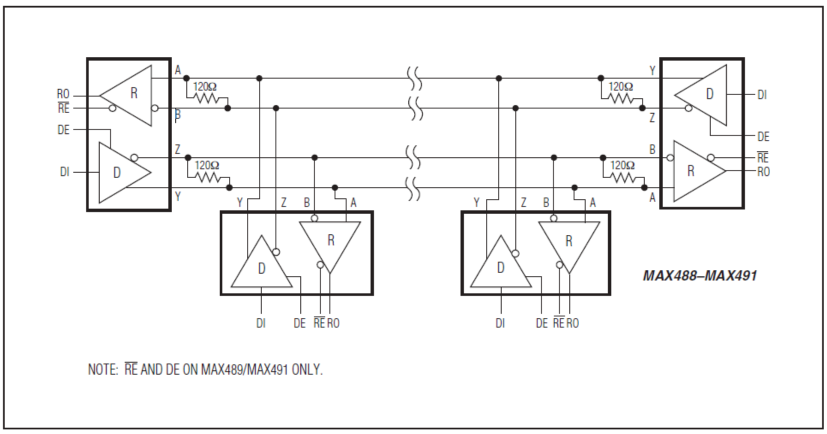


Figure : RS485 output signal

MAX489

The MAX489 chips is a low-power transceiver for RS-485 and RS422 communication. Each chip contains one driver and one receiver. The chip is powered by a 5V supply voltage. The MAX489 features reduced slew-rate drivers that minimize EMI and reduce reflections caused by improperly terminated cables, thus allowing error-free data transmission up to 250kbps. The chip is powered by a 5V supply voltage.2





11. Annexure C: EMI technology

TVS diodes

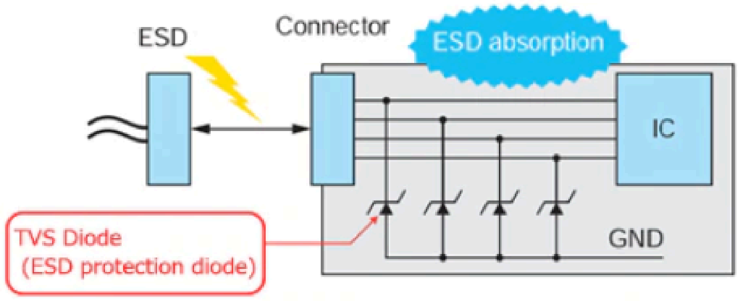


Fig. 2-5(a) Example of usage of TVS diodes

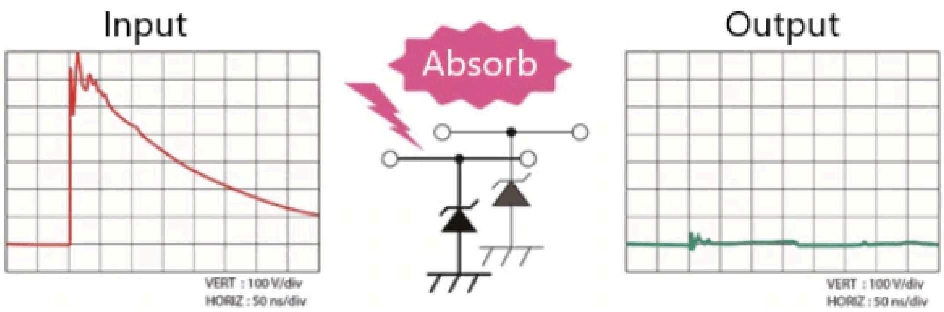


Figure : TVS diode implementation