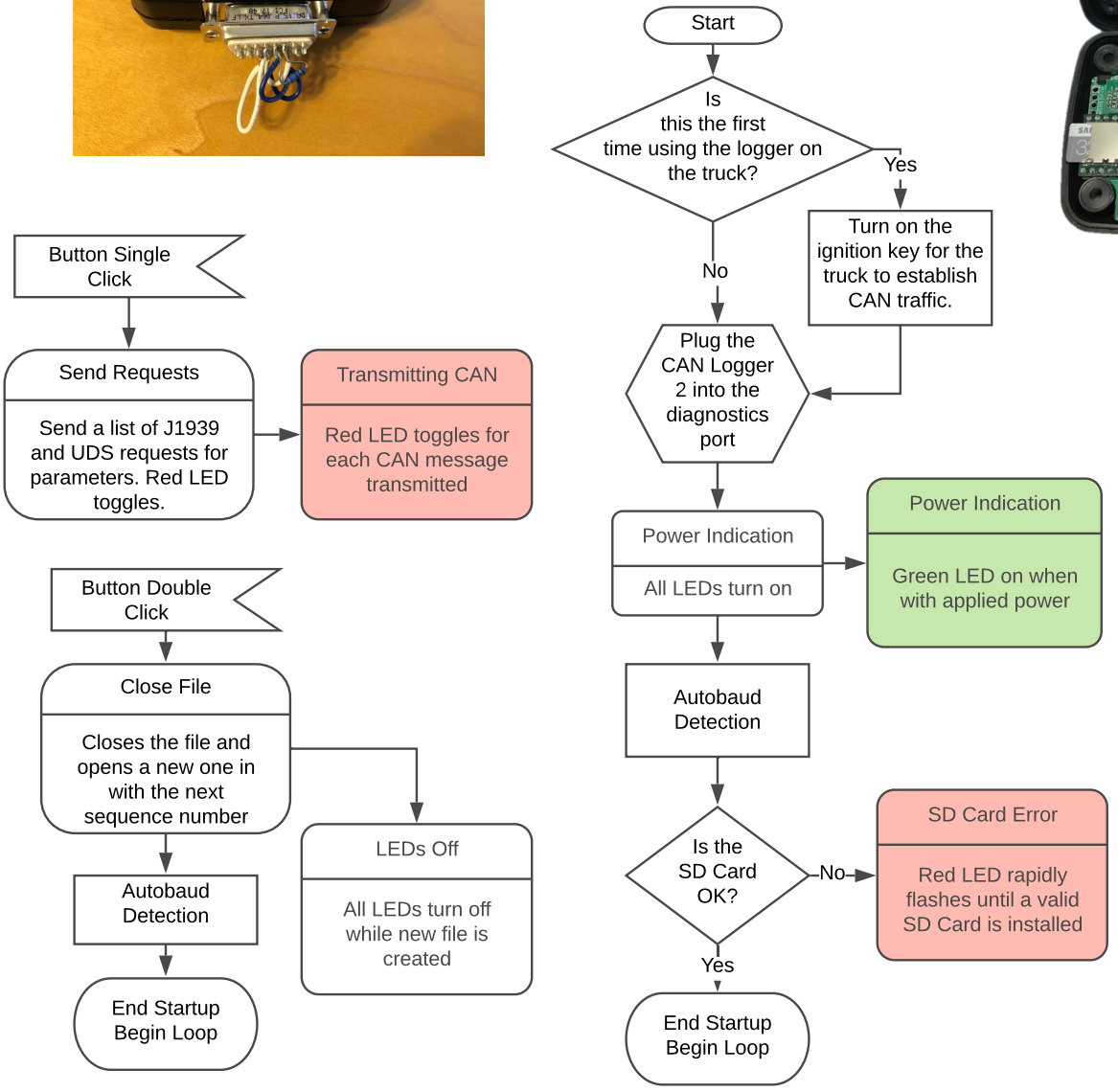
National Science Foundation

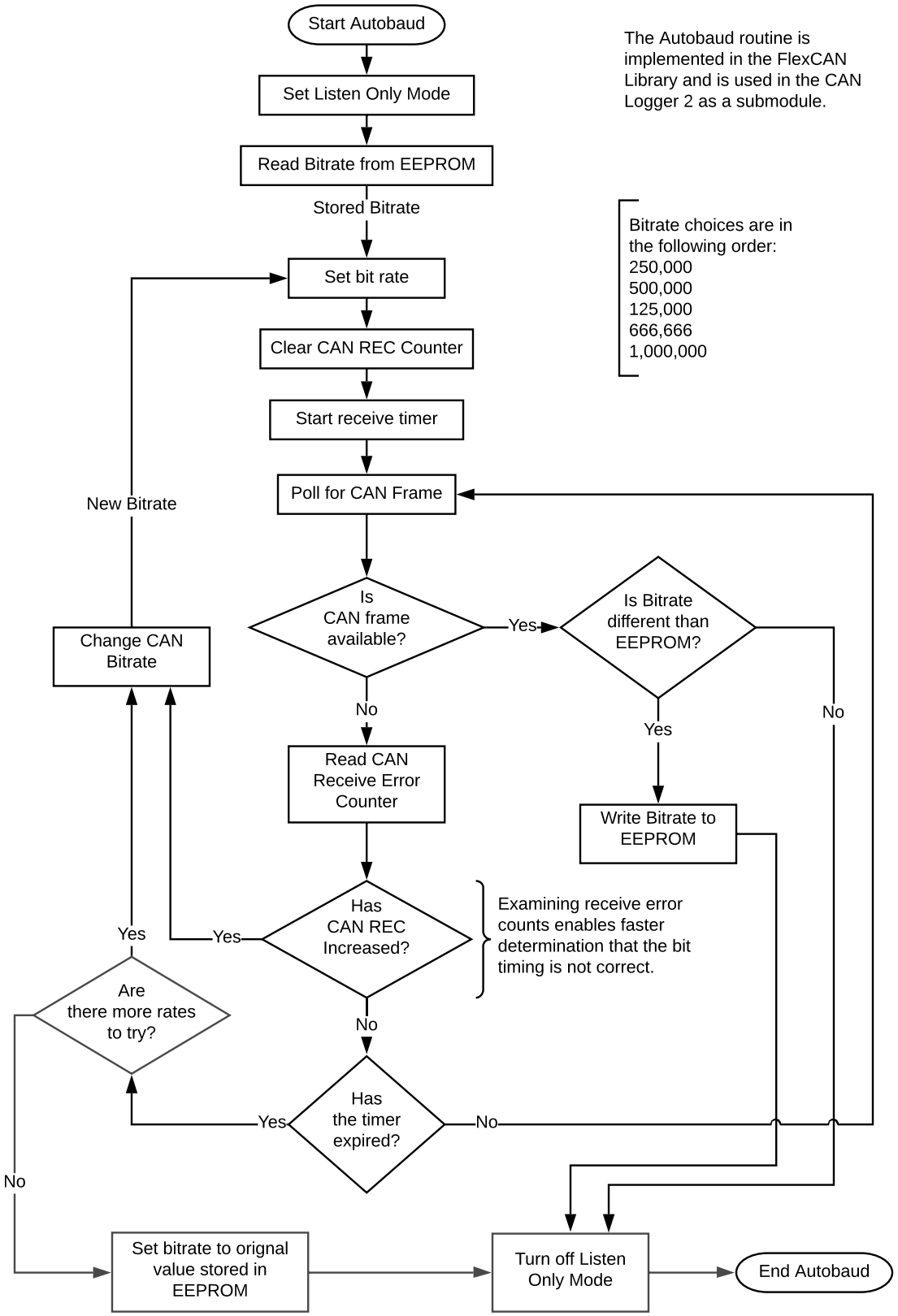
THE UNIVERSITY of TULSA
Department of Mechanical Engineering

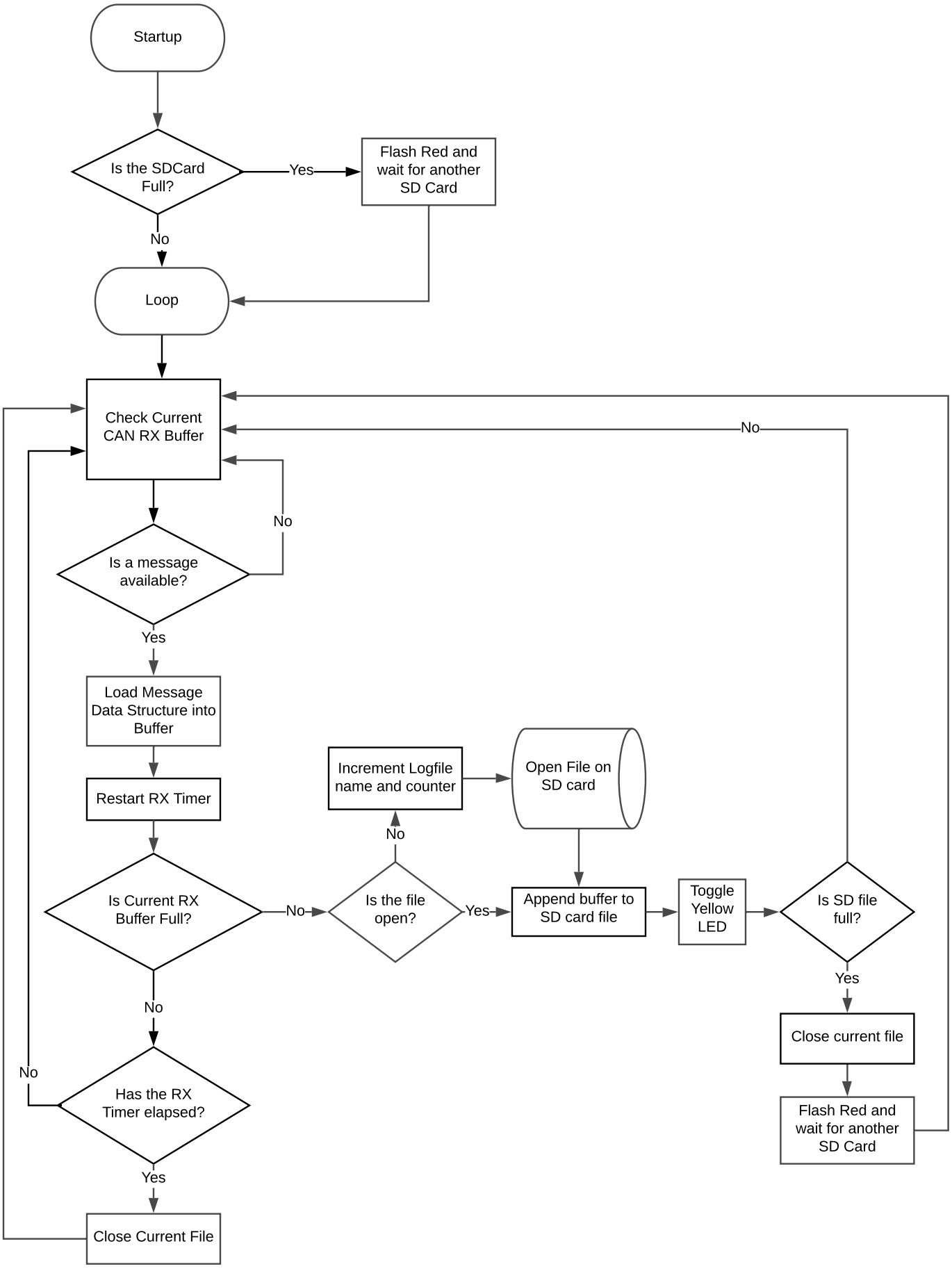
NMFTA
National Motor Freight Traffic Association, Inc.

CAN Logger 2



This material is based upon work supported by the National Science Foundation under Grant No. 1715409, titled "Detecting and Reconstructing Network Anomalies and Intrusions in Heavy Duty Vehicles"

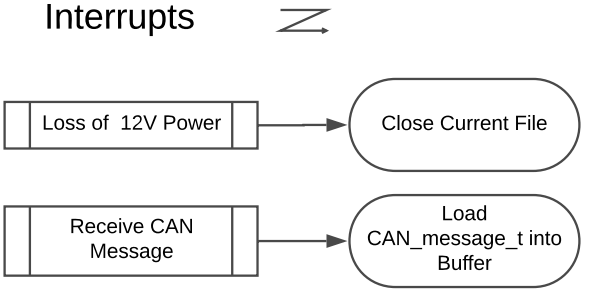

Start Autobaud
Set Listen Only Mode
Read Bitrate from EEPROM
Stored Bitrate
Set bit rate
Clear CAN REC Counter
Start receive timer
Poll for CAN Frame
New Bitrate
Is CAN frame available?
Yes
Is Bitrate different than EEPROM?
No
Yes
No
Change CAN Bitrate
Read CAN Receive Error Counter
Write Bitrate to EEPROM
Has CAN REC Increased?
Yes
Yes
No
Are there more rates to try?
Yes
Has the timer expired?
Yes
No
No
Set bitrate to orignal value stored in EEPROM
Turn off Listen Only Mode
End Autobaud
The Autobaud routine is implemented in the FlexCAN Library and is used in the CAN Logger 2 as a submodule.
Bitrate choices are in the following order:
250,000
500,000
125,000
666,666
1,000,000
Examining receive error counts enables faster determination that the bit timing is not correct.


Startup
Is the SDCard Full?
Yes
Flash Red and wait for another SD Card
No
Loop
Check Current CAN RX Buffer
No
Is a message available?
No
Yes
Load Message Data Structure into Buffer
Restart RX Timer
Increment Logfile name and counter
Open File on SD card
No
Is Current RX Buffer Full?
No
Is the file open?
Yes
Append buffer to SD card file
Toggle Yellow LED
Is SD file full?
Yes
Close current file
Flash Red and wait for another SD Card
No
Has the RX Timer elapsed?
No
Yes
Close Current File

Serial Commands

- HEX
- BIN
- DEL
- STOP
- START
- LS
- LS -A
- FORMAT
- BAUD
- ERRORS
- STREAM ON
- STREAM OFF
- baudRate
- {filename}
- help

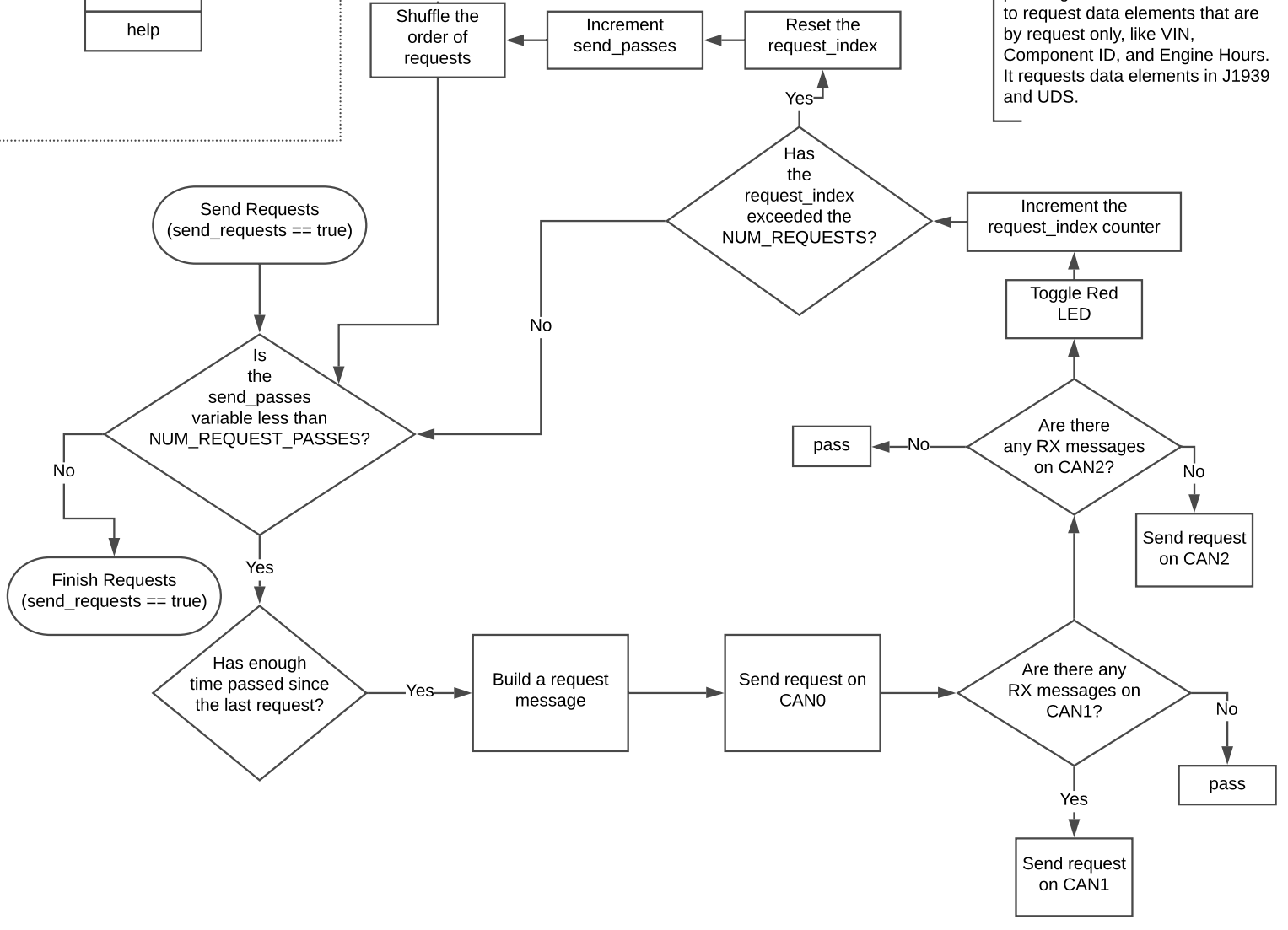
Interrupts

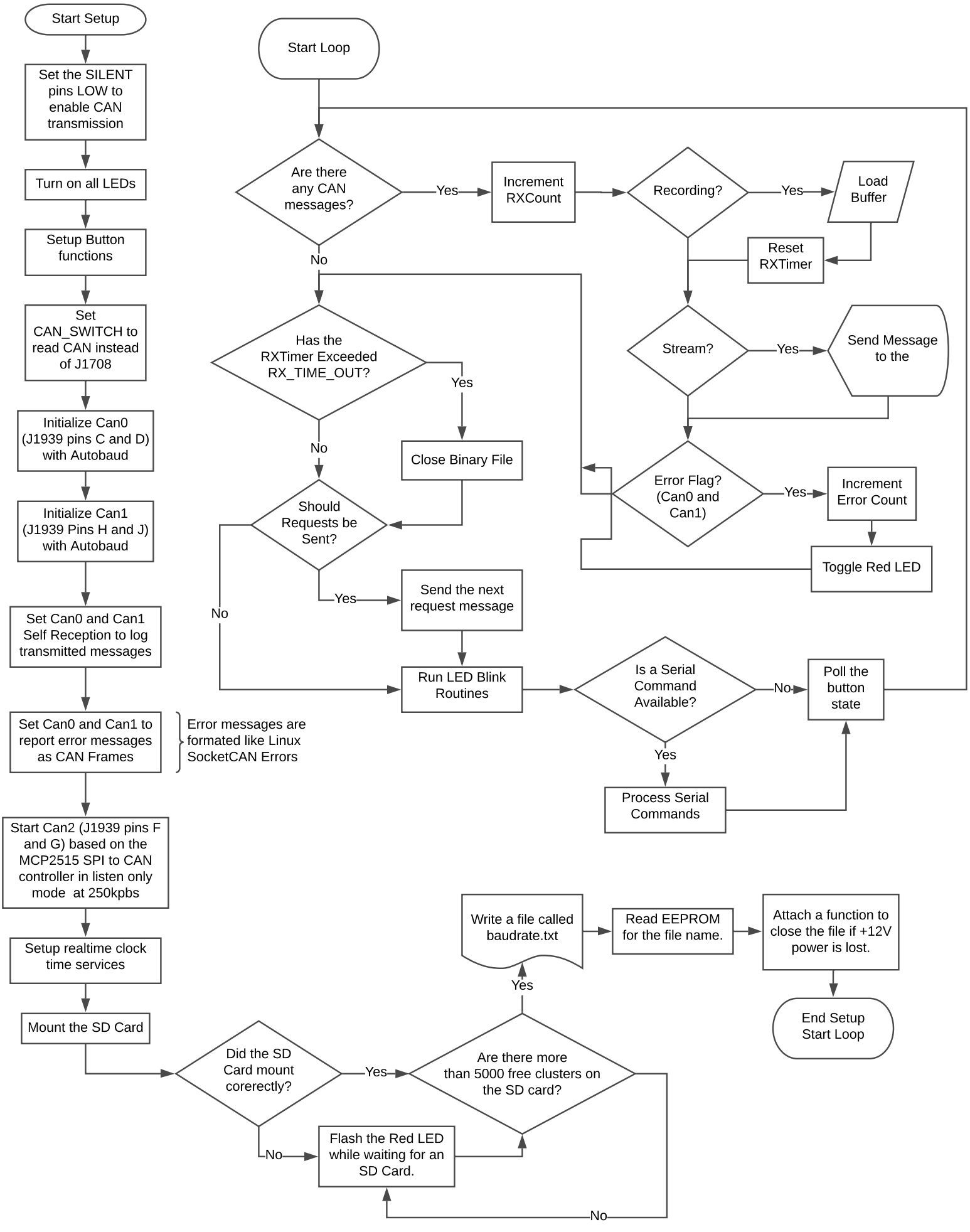


Shuffling the requests changes the order and can help extract additional data. If two long messages are reuqested back to back, one of the messages may not come through. By reordering, there is a better chance the message that didn't come through is available.

Sending Requests

This routine is user activated by pressing the button. It is intended to request data elements that are by request only, like VIN, Component ID, and Engine Hours. It requests data elements in J1939 and UDS.




Start Setup
Set the SILENT pins LOW to enable CAN transmission
Turn on all LEDs
Setup Button functions
Set CAN_SWITCH to read CAN instead of J1708
Initialize Can0 (J1939 pins C and D) with Autobaud
Initialize Can1 (J1939 Pins H and J) with Autobaud
Set Can0 and Can1 Self Reception to log transmitted messages
Set Can0 and Can1 to report error messages as CAN Frames
Error messages are formated like Linux SocketCAN Errors
Start Can2 (J1939 pins F and G) based on the MCP2515 SPI to CAN controller in listen only mode at 250kpbs
Setup realtime clock time services
Mount the SD Card
Did the SD Card mount corerectly?
Yes
No
Flash the Red LED while waiting for an SD Card.
Are there more than 5000 free clusters on the SD card?
Yes
No
Write a file called baudrate.txt
Read EEPROM for the file name.
Attach a function to close the file if +12V power is lost.
End Setup Start Loop
Start Loop
Are there any CAN messages?
Yes
No
Increment RXCount
Recording?
Yes
Load Buffer
Reset RXTimer
Has the RXTimer Exceeded RX_TIME_OUT?
Yes
No
Close Binary File
Should Requests be Sent?
Yes
No
Send the next request message
Stream?
Yes
Send Message to the
Error Flag? (Can0 and Can1)
Yes
Increment Error Count
Toggle Red LED
Run LED Blink Routines
Is a Serial Command Available?
No
Yes
Poll the button state
Process Serial Commands

SD Card Memory Block. 512 Bytes are stored at a time in the following format

Bytes	0	1	2	3	4 through 478	479	480	481	482	483	484	485	486	487	488	489	490
Data	C	A	N	2	Nineteen (19) CAN Frames	RXCount0				RXCount1				RXCount2			
Hex	43	41	4E	32	SEE CAN FRAME STRUCTURE	MSB			LSB	MSB			LSB	MSB			LSB
Notes	Characters					uint32_t				uint32_t				uint32_t			

491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511
Can0	Can1	Can2	Can0	Can1	Can2	T	U	2	_	_	N1	N2	N3	Write Time			CRC32			
uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	54	55	32			ASCII Encoded			MSB		LSB	MSB			LSB
Receive Error Counts			Transmit Error Counts			Version			Logger Number		File Number			Microseconds for SDCard			Calculated from bytes 0 through 507			

CAN Frame Structure

Bytes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Data	Current	Timestamp				System				CAN Identifier				DLC	Microseconds per			B0	B1	B2	B3	B4	B5	B6	B7
Hex	0 | 1 | 2	LSB			MSB	LSB			MSB	LSB			MSB	8	LSB		MSB	01	02	03	04	05	06	07	08
Notes	Corresponds to Can0, Can1, or Can2	Number of seconds from the epoch (1970)				The system microsecond counter when the CAN registers were read.				CAN ID with the Error Flags and Extended Flag, like Socket CAN				Data Length Code	Fractional seconds per tick of the Timestamp			Message Data Bytes padded with x0FF if not used.							

EEPROM Memory Map

0x00	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Data	Bitrate	Bitrate	Bitrate	RES	2	_	_	null	N1	N2	N3	null	T	U	null
Hex					32			0x00	ASCII Encoded						0x00
Notes	Can0 Bitrate	Can1 Bitrate	Can2 Bitrate		Logger Identifier of 2 uppercase letters				File ID. Each digit can be 0-9 or A-Z for a total of 36*3 = 46,656 files.				Brand Name of Logger (i.e. "TU") to start each filename.		

CAN_message_t

```
uint32_t id;          // can identifier
uint32_t micros;      // system microseconds
uint32_t rxcount;     // number of received messages
uint16_t timestamp;   // FlexCAN time when message arrived
struct {
  uint8_t extended:1; // identifier is extended (29-bit)
  uint8_t remote:  1; // remote transmission request packet type
  uint8_t overrun: 1; // message overrun
  uint8_t reserved:5;
} flags;
uint8_t len;          // length of data
uint8_t buf[8];       // data bytes
```