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
```