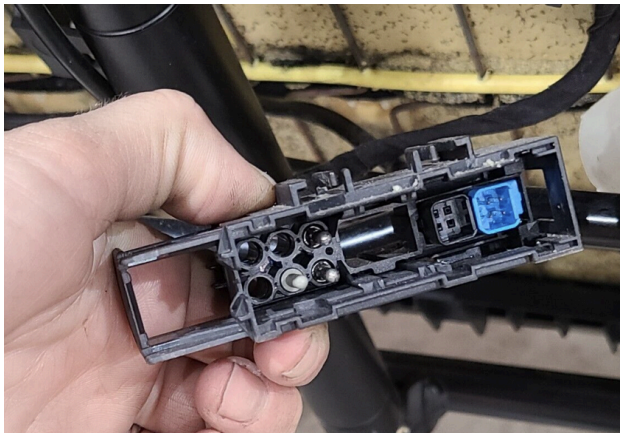
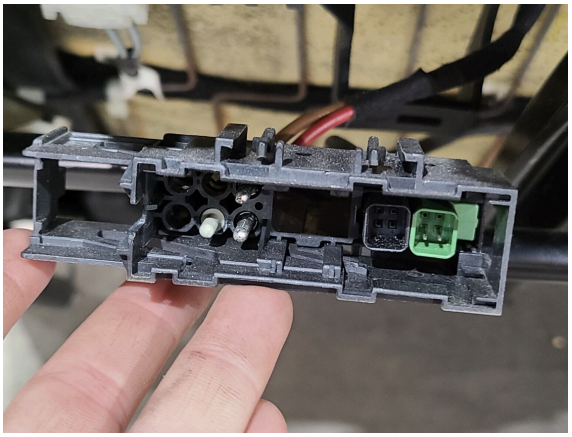
Goal: Replace 996/986 Seats with 997/987 Seats

996/986 Seats Technical Info

996/986 Seat Connectors



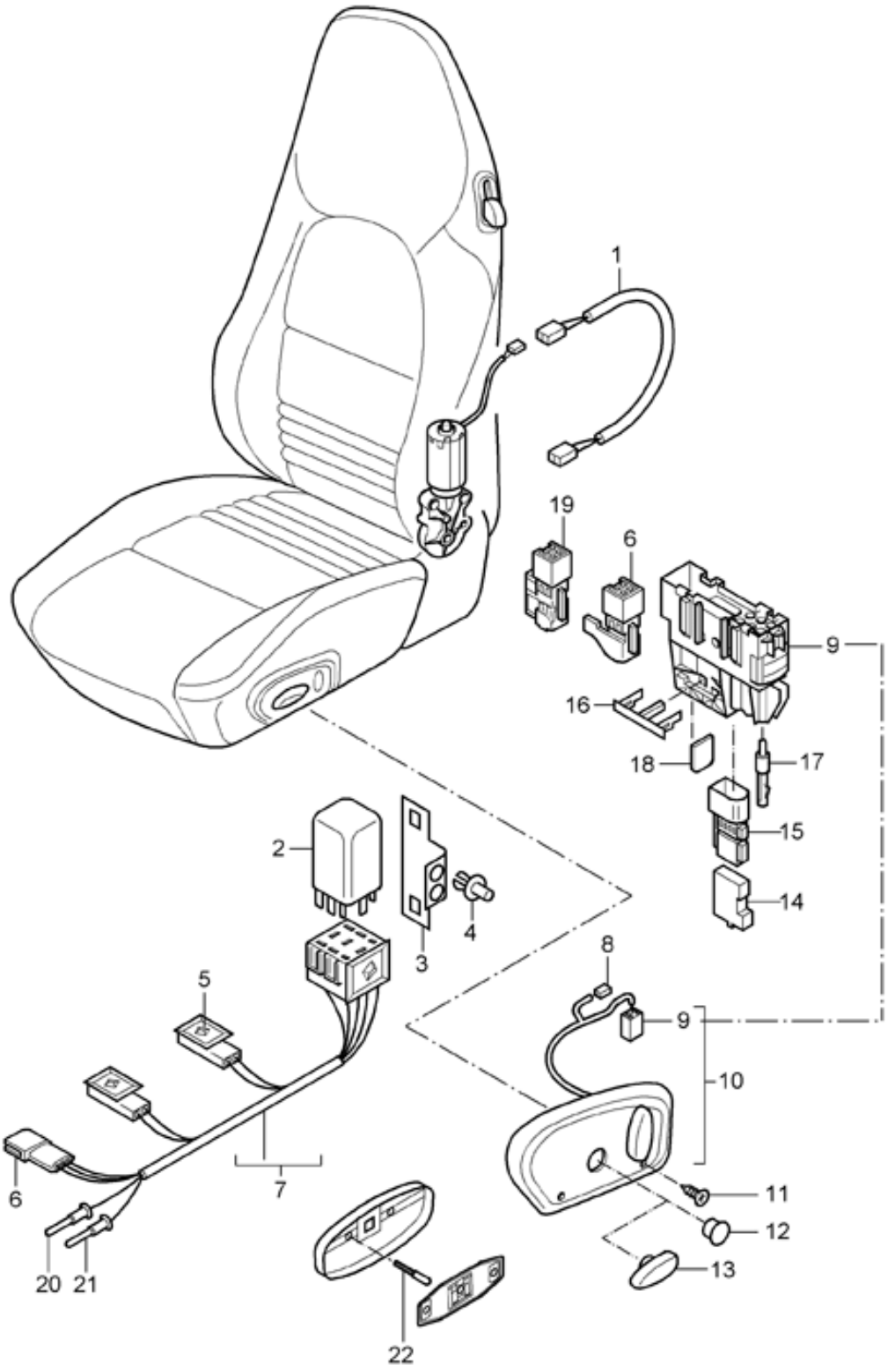


986 Wiring Diagram

- 1 Round pin is power
- 1 Round pin is ground
- 1 Round pin is a plastic alignment pin
- Square black connector are the seat adjustment buttons
- Blue (left) and Green (right) are the seatbelt connections (is seatbelt fastened)
- The heated seat pins are not populated in this photo but occupy unused round pin locations.

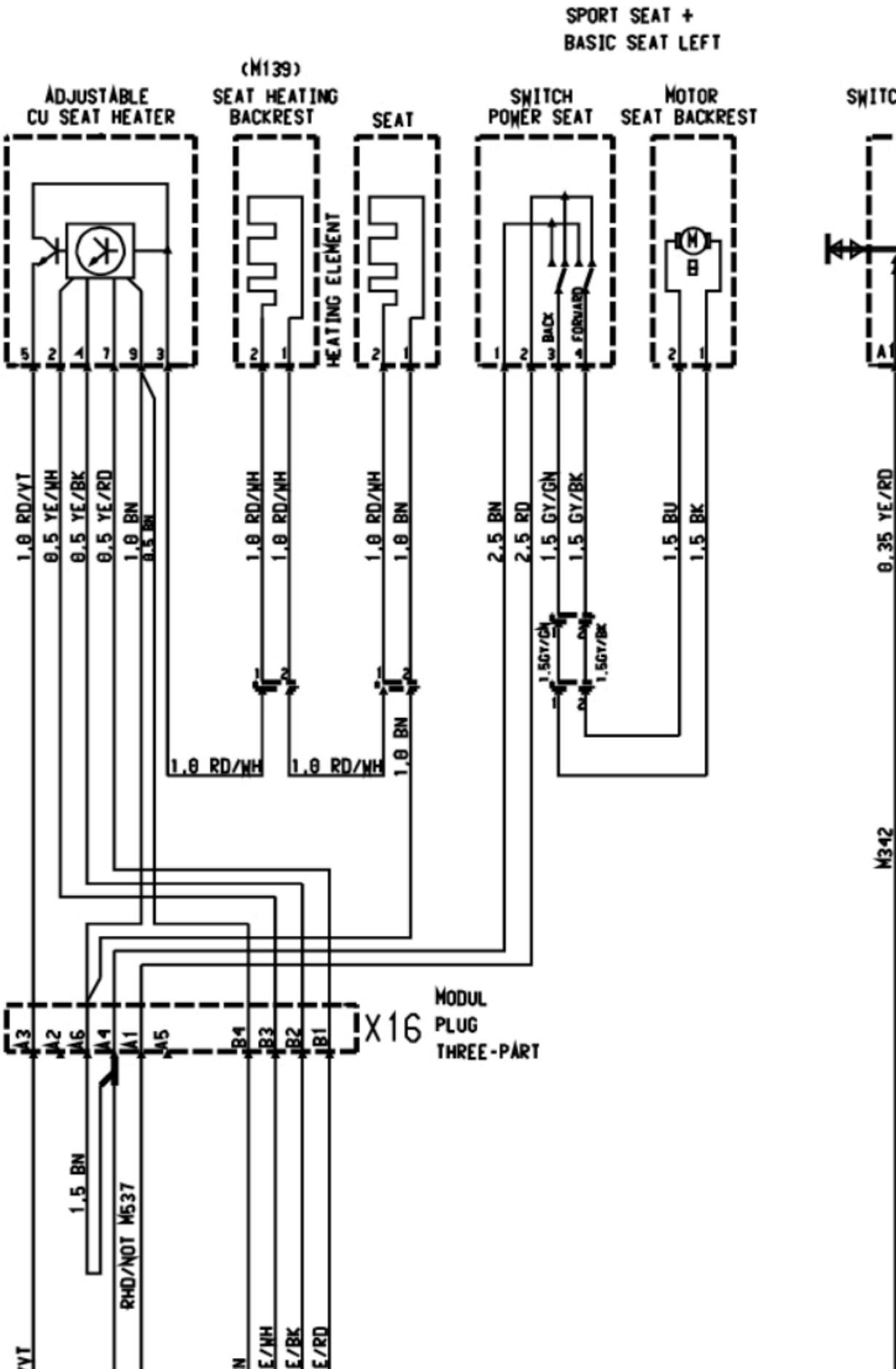
Note that 996/986 seats do not have airbags in the seat itself.

996/986 Seat Diagram



Pos	Part Number	Description	Qty. Needed	Price	Year	Which Seat?
1	99661254300	wiring harness for backrest adjustment	2	$28.12	1997-2004	Driver / Passenger
2	99661852100	control unit seat heater	2	$233.51	1997-2004	Driver / Passenger
3	99661853100	support control unit	2	$19.65	1997-2004	Driver / Passenger
4	99959171240	expansion rivet 6,0 x 12,0	2	$1.10	1997-2002	Driver / Passenger
-4	90014319102	tapping screw	4	$1.28	2002-2004	Driver / Passenger
5	99965274840	pin connector socket 2-pole seat heater	2	$1.76	1997-2004	Driver / Passenger
6	99965021640	pin connector socket 4-pole black to seat heather button	1	$7.67	1997-2004	Driver / Passenger
7	99661254001	wiring harness seat heater	2	$163.47	1997-2004	Driver
8	99965274740	pin connector socket 2-pole for backrest adjustment	1	$1.76	1997-2004	Driver / Passenger
9	99965018040	pin connector socket 6-pole wiring harness passenger compartment	2	$15.88	1997-2004	Driver
10	99661371300EBO	seat adjustment black/black-grey	1	$706.92	1997-2004	Driver
10	99661371401EJM	seat adjustment switch grey/black		$706.92	1999-2004	Driver / Passenger
10	99661371401A03	seat adjustment switch black		$706.92	1997-2004	Driver / Passenger
11	99991917302	countersunk-head screw 3,0 x 18	4	$1.28	1997-2004	Driver / Passenger
12	9966137110001C	dummy plug switch		$1.20	1997-2004	Driver / Passenger
13	98652162700	handle height adjustment	1	$30.34	1997-2004	Driver
-13	98652162800	handle height adjustment	1	$30.34	1997-2004	Passenger
14	99965024840	cap	2	$2.31	1997-2004	Driver / Passenger
15	99965025040	pin connector socket 8-pole	2	$1.69	1997-2004	Driver / Passenger
16	99965024740	slide 6-pole	2	$2.31	1997-2004	Driver / Passenger
17	99965022540	pin 2,5	2	$2.07	1997-2004	Driver / Passenger
18	99965296040	lid	2	$1.10	1997-2004	Driver / Passenger
19	99965021840	pin connector socket 4-pole white	2	$8.18	1997-2004	Driver / Passenger
20	99965257022	connector 0.5 - 1.0 qmm	X	$1.45	1997-2004	Driver / Passenger
21	99965257122	connector 1.0 - 2.5 qmm	X	$1.28	1997-2004	Driver / Passenger

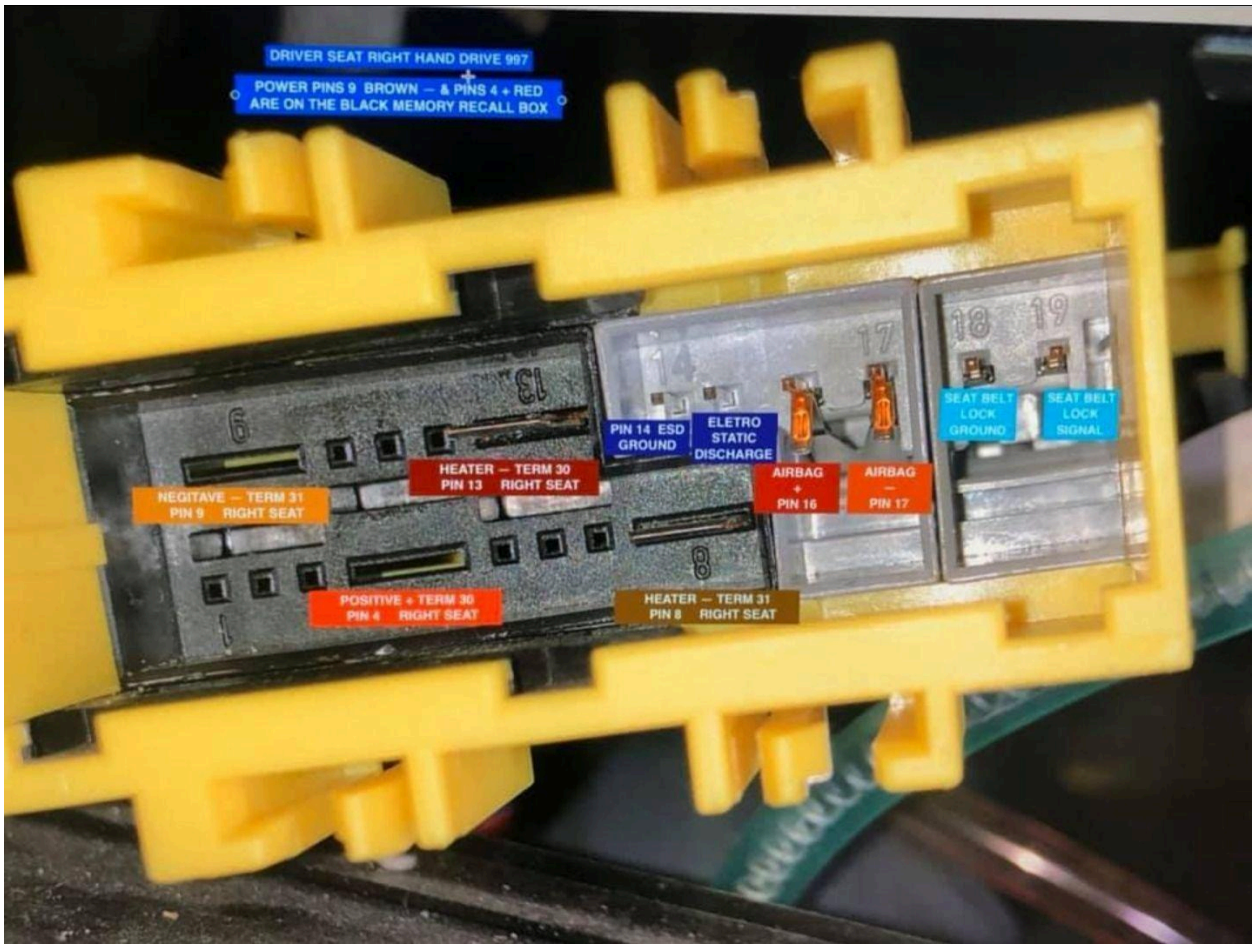
996/986 Seat Wiring

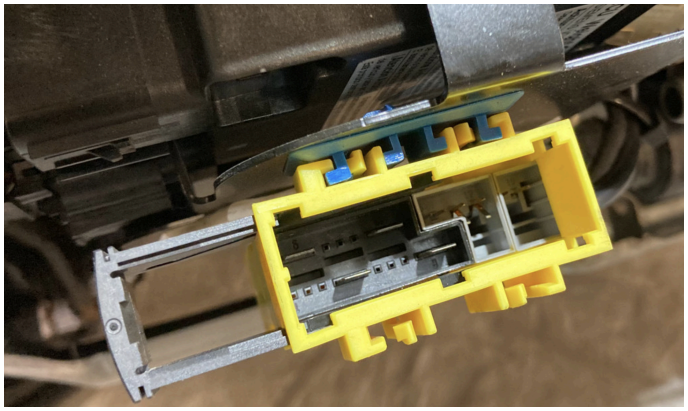

SPORT SEAT +
BASIC SEAT LEFT
(M139)
ADJUSTABLE CU SEAT HEATER
SEAT HEATING BACKREST
SEAT
SWITCH POWER SEAT
MOTOR SEAT BACKREST
HEATING ELEMENT
BACK
FORWARD
1,0 RD/VT
0,5 YE/WH
0,5 YE/BK
0,5 YE/RD
1,0 BN
0,5 BK
1,0 RD/WH
2,5 BN
2,5 RD
1,5 GY/GN
1,5 GY/BK
1,5 BU
1,5 BK
X16
MODUL PLUG THREE-PART
1,5 BN
RHD/NOT M537
0,35 YE/RD
M342

Wiring Diagrams and Info for heated seats
https://www.dennisvogel.com/heatedseats/

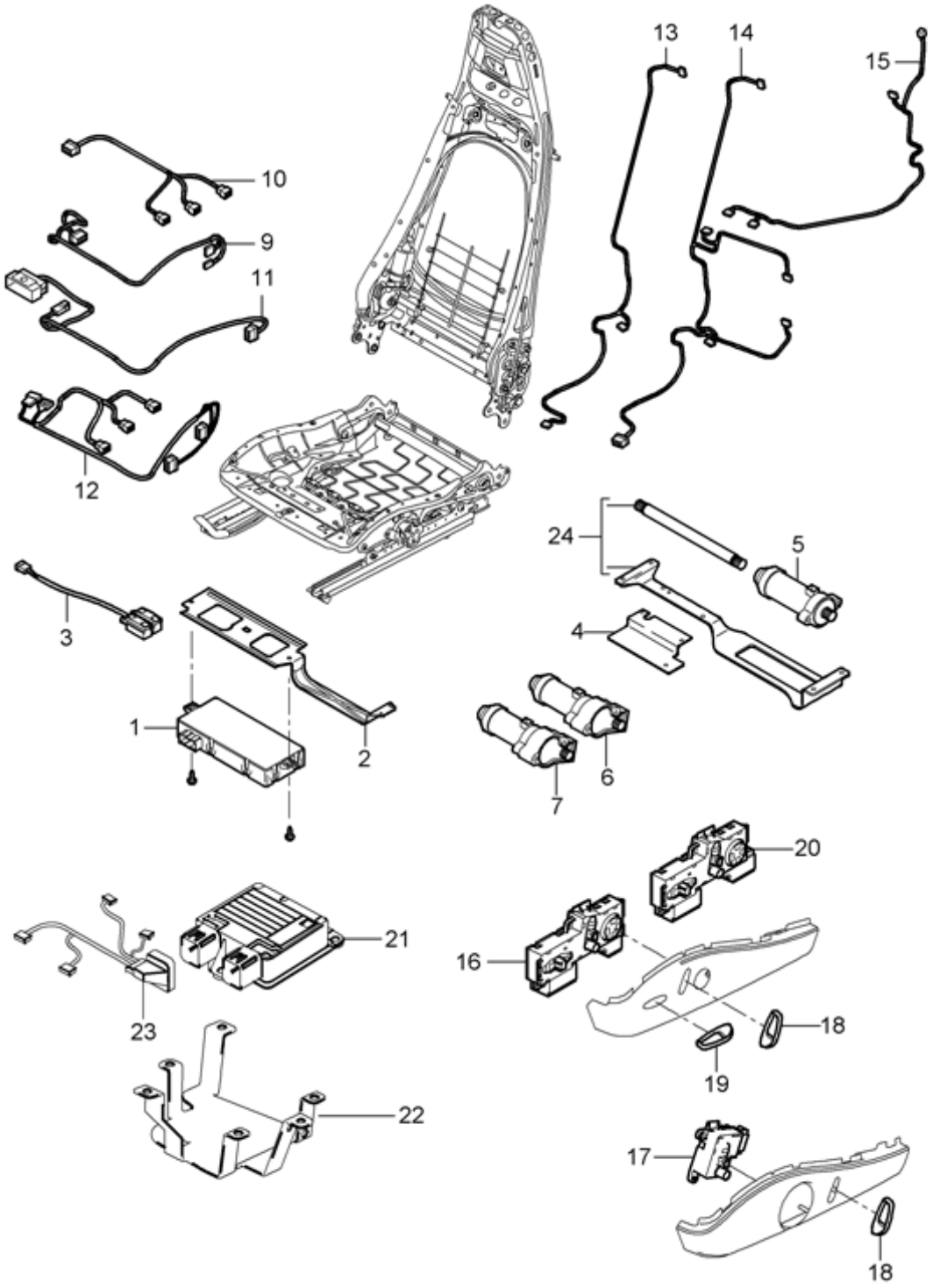
997/987 Seats

997/987 Passenger Seat





997/987 Seat Diagram

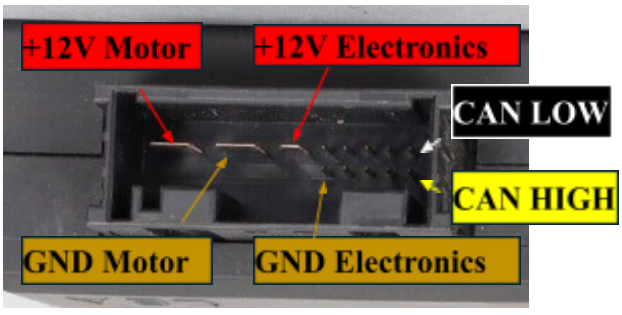


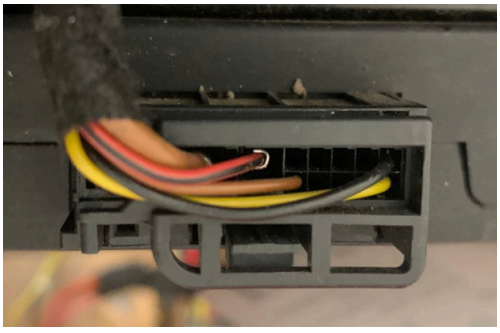
Pos	Part Number	Description	Qty.	Price	Year	Which Seat?
1	99761853707	control unit seating position control	1	$803.48	2005-2008	Driver
2	99752164301	bracket - control unit seating position control	1	$50.93	2005-2008	Driver
3	99761372500V05	Seat Adj Switch - Galvano Silver	1	$765.84	2005-2008	Driver
-3	99761372600V05	Seat Adj Switch - Galvano Silver	1	$765.84	2005-2008	Passenger
4	99752134300	support electric motor	1	$18.92	2005-2008	Driver / Passenger
5	99762432200	electric motor seat adjuster lengthways without memory	1	$435.27	2005-2008	Driver / Passenger
-5	99762433200	electric motor seat adjuster lengthways with memory	1	$456.19	2005-2008	Driver / Passenger
6	99762432200	electric motor seat adjuster height adjustment without memory	1	$435.27	2005-2008	Driver / Passenger
-6	99762433200	electric motor seat adjuster height adjustment with memory	1	$456.19	2005-2008	Driver / Passenger
7	99762432300	electric motor seat adjuster tilt angle without memory	1	$435.27	2005-2008	Driver / Passenger
-7	99762433300	electric motor seat adjuster tilt angle with memory	1	$456.19	2005-2008	Driver / Passenger
9	99761268400	wiring harness switch seat adjuster memory	1	$156.72	2005-2008	Driver / Passenger
10	99761269000	wiring harness height adjustment lengthways memory	1	$182.67	2005-2008	Driver / Passenger
11	99761268201	wiring harness - switch	1	$156.72	2005-2008	Driver / Passenger
12	99761269101	wiring harness switch without memory	1	$182.67	2005-2007	Driver / Passenger
12	99761269102	wiring harness switch without memory	1	$182.67	2008	Driver / Passenger
13	99761268900	wiring harness lumbar support without memory	1	$182.67	2005-2008	Driver / Passenger
14	99761269201	wiring harness backrest lumbar with memory	1	$213.19	2005-2008	Driver / Passenger
15	99761268501	wiring harness - without memory	1	$86.47	2005-2008	Driver
-15	99761268601	wiring harness - without memory	1	$86.47	2005-2008	Passenger
16	99761372101V05	switch seat adjuster memory switch	1	$475.44	2005-2008	Driver
-16	99761372201V05	switch seat adjuster memory switch	1	$475.44	2005-2008	Passenger
17	99761371301	switch seat adjuster manually adjustable	1	$852.21	2005-2008	Driver
-17	99761371401	switch seat adjuster manually adjustable	1	$852.21	2005-2008	Passenger
18	99752163101V05	control button, backrest - galvano silver	1	$50.26	2005-2008	Driver / Passenger
19	99752163303V05	control button, Seat Cushion	1	$50.26	2005-2008	Driver / Passenger
20	99761371702V05	switch seat adjuster without memory	1	$765.84	2005-2008	Driver
-20	99761371801V05	switch seat adjuster without memory	1	$765.84	2005-2008	Passenger
21	99761823305	control unit - seat occupancy detection	1	$522.25	2005-2008	Passenger
23	99761268801	wiring harness - seat occupancy detection	1	$152.19	2006-2008	Passenger
-24	99752190100	repair kit pliable shaft engine bracket	1	$203.16		
-	99965097340	clip mount wiring harness seat occupancy detection	1	$1.33	2005-2008	Driver / Passenger

997/987 Driver Seat

#1 Seating Position Control Unit. 997.618.537.07

+ Power Connector, front center of seat (bottom right in the above picture)





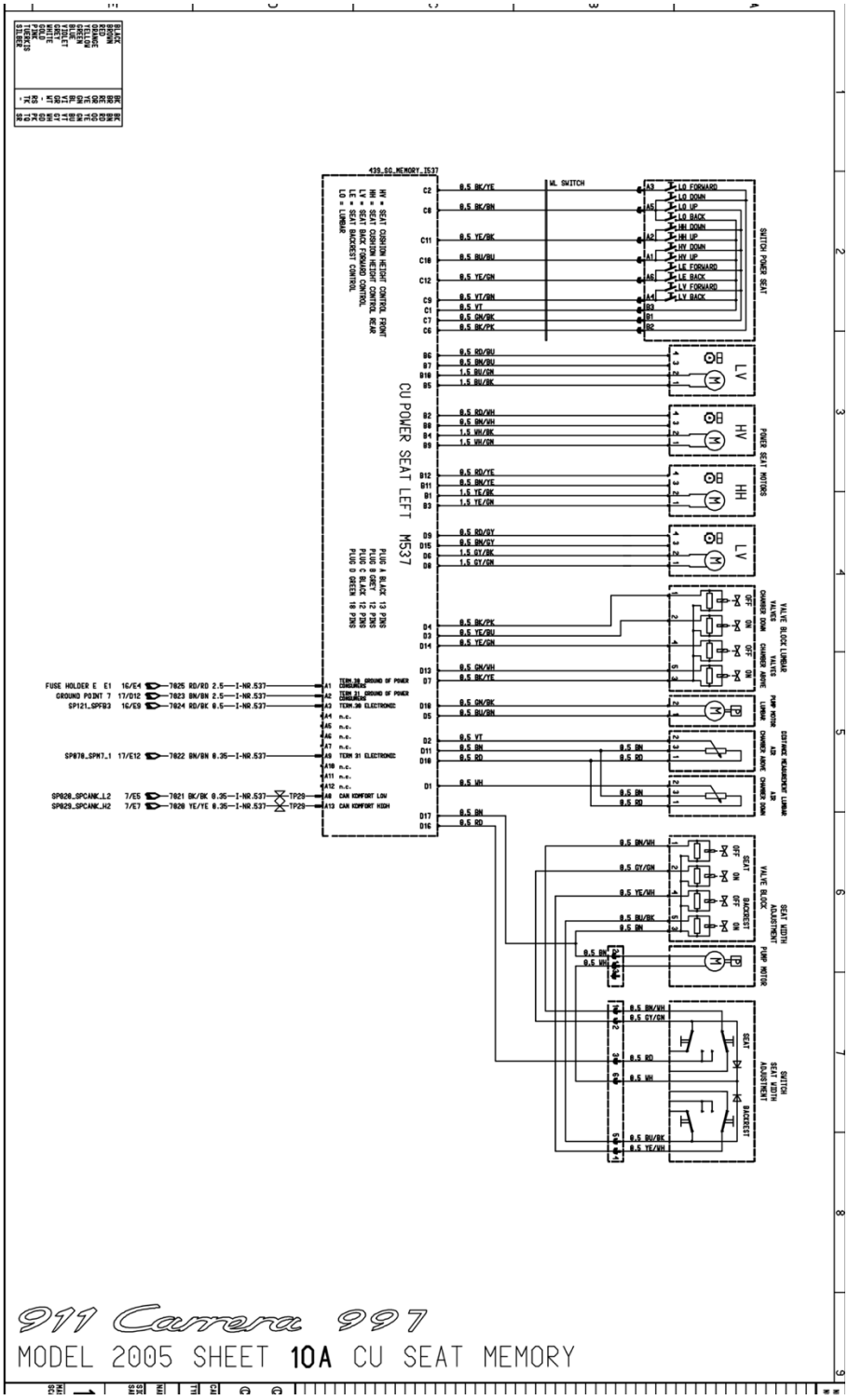
The above two pictures show the “A” connector unplugged (left) and plugged in (right).

Only 3 spade connections and 3 pins are used per the wiring diagram below.
Power comes from the car’s power while CAN is ‘Comfort CAN’ from the Gateway CU

I need traffic on these CAN pins in order to see ‘Comfort CAN’ traffic…

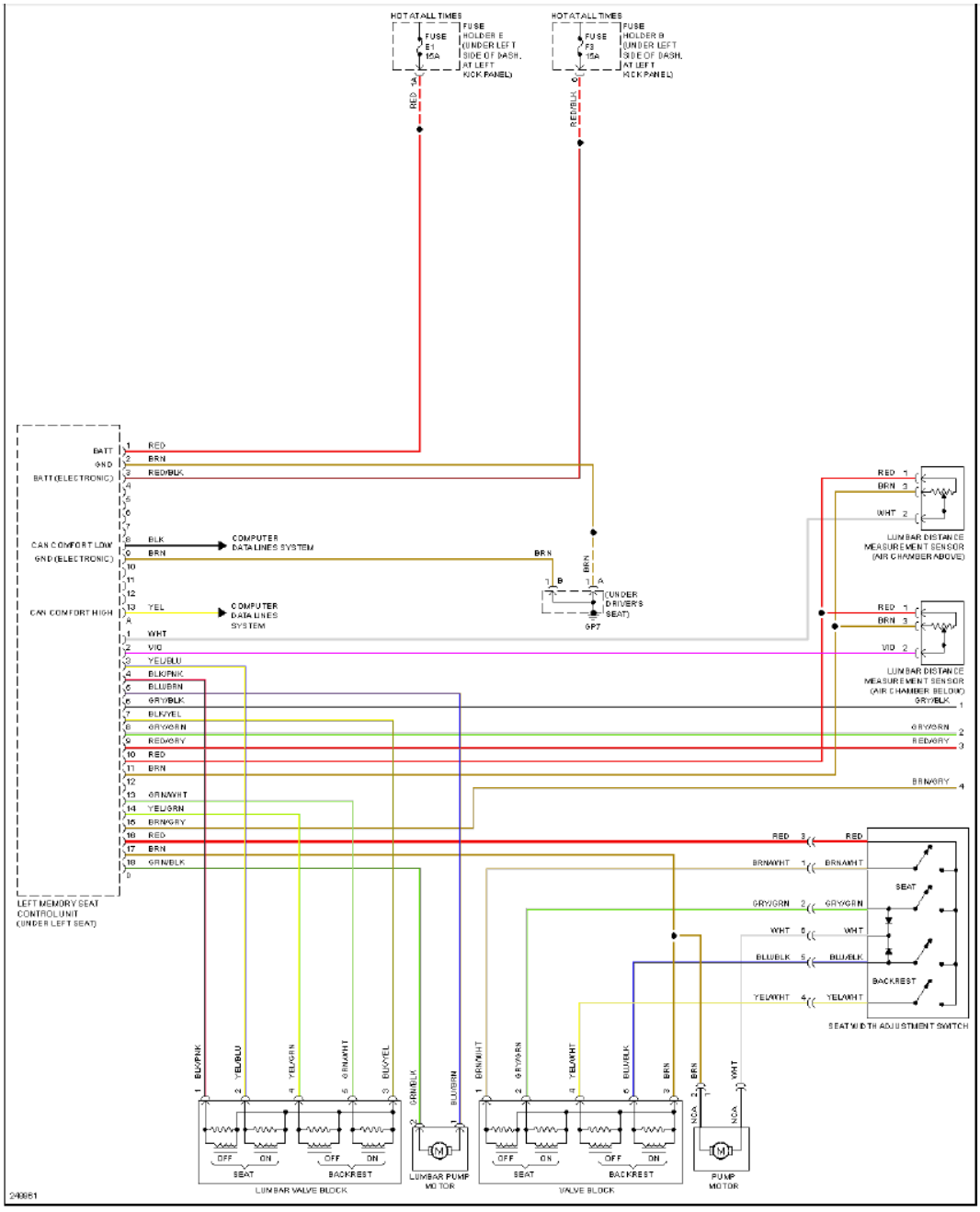
#1 Seating Position Control Unit Wiring

This first diagram seems to be the ‘official’ wiring diagram from the shop manual. (But the wiring names are not clear.) Second is much more clear but less official.



911 Carrera 997

MODEL 2005 SHEET **10A** CU SEAT MEMORY

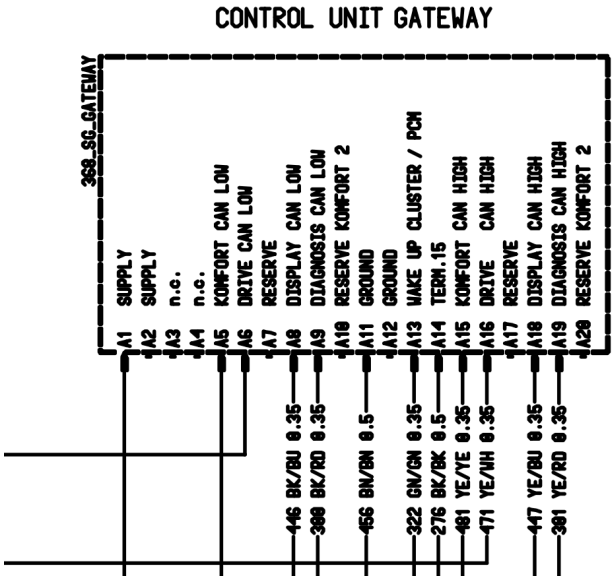

HOT AT ALL TIMES
FUSE E1 15A
FUSE HOLDER E (UNDER LEFT SIDE OF DASH, AT LEFT KICK PANEL)
HOT AT ALL TIMES
FUSE F3 15A
FUSE HOLDER B (UNDER LEFT SIDE OF DASH, AT LEFT KICK PANEL)
RED 14
RED/BLK 3
BATT
GND
BATT (ELECTRONIC)
CAN COMFORT LOW
GND (ELECTRONIC)
CAN COMFORT HIGH
RED
BRN
RED/BLK
BLK
COMPUTER DATA LINES SYSTEM
BRN
YEL
COMPUTER DATA LINES SYSTEM
WHT
VIO
YEL/BLU
BLU/PNK
BLU/BRN
GRY/BLK
BLK/YEL
GRY/GRN
RED/GRY
RED
BRN
GRN/WHT
YEL/GRN
BRN/GRY
RED
BRN
GRN/BLK
LEFT MEMORY SEAT CONTROL UNIT (UNDER LEFT SEAT)
BRN
(UNDER DRIVER'S SEAT)
6P7
LUMBAR DISTANCE MEASUREMENT SENSOR (AIR CHAMBER ABOVE)
LUMBAR DISTANCE MEASUREMENT SENSOR (AIR CHAMBER BELOW)
GRY/BLK
GRY/GRN
RED/GRY
BRN/GRY
SEAT
BACKREST
SEAT WIDTH ADJUSTMENT SWITCH
BRN/WHT
GRY/GRN
WHT
BLU/BLK
YEL/WHT
BLU/PNK
YEL/BLU
YEL/GRN
GRN/WHT
BLU/YEL
GRN/BLK
BLU/BRN
BRN/WHT
GRY/GRN
YEL/WHT
BLU/BLK
BRN
NCA
WHT
OFF
ON
SEAT
BACKREST
LUMBAR VALVE BLOCK
LUMBAR PUMP MOTOR
VALVE BLOCK
PUMP MOTOR
249861

Control Unit Gateway

The CU Gateway connects all of the CAN networks together and to the Debug OSBII (Diagnosis) port. On my bench I can connect an ODBII to Diagnosis and query the Seat Memory module through the gateway.

CU Gateway 997.610.107.04

BMA



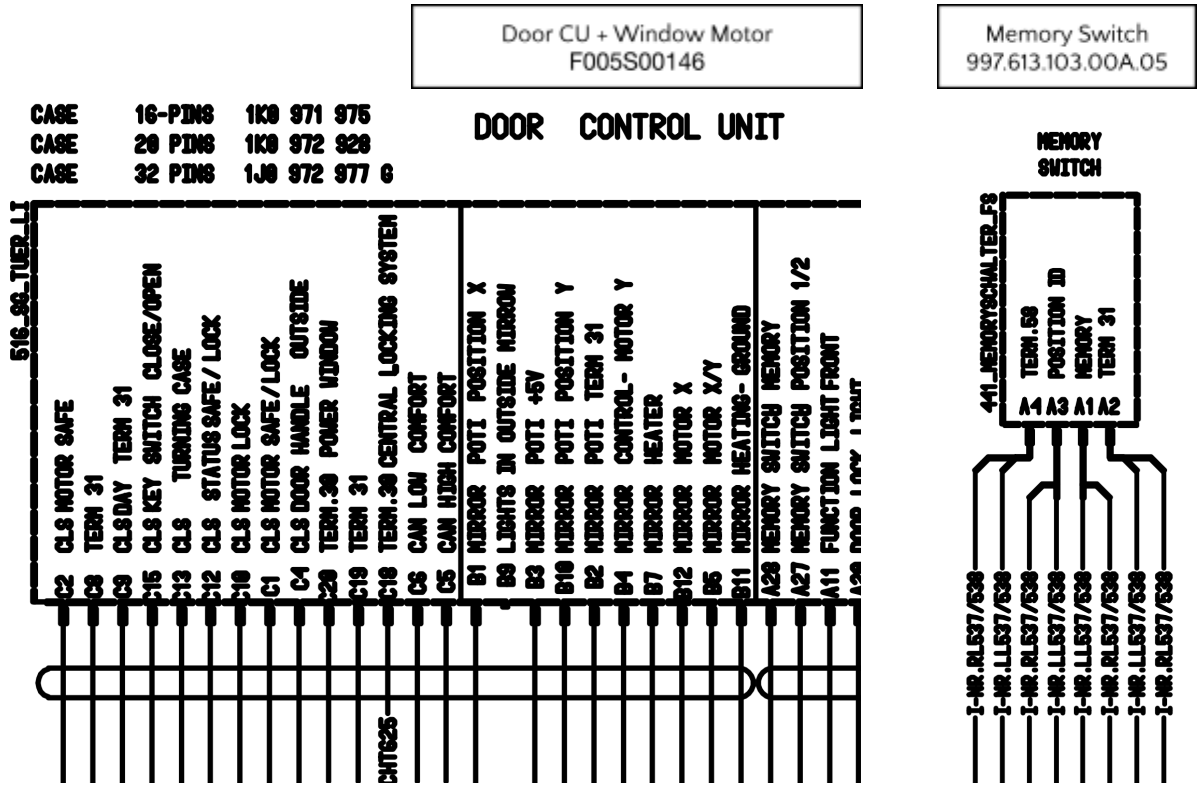
A1	Supply (+12V)	RD/GN	**A11**	Ground	BN/BN
A2	n.c. (Supply)	NONE	A12	n.c. (Ground)	NONE
A3	n.c.	NONE	A13	Wake-Up Cluster / PCM	GN/GN
A4	n.c.	NONE	**A14**	Term.15 (Ignition +12V)	BK/BK
A5	Komfort CAN LOW	BK/BK	**A15**	Komfort CAN HIGH	YE/YE
A6	Drive CAN LOW	BK/WH	A16	DRIVE CAN HIGH	YE/WH
A7	Reserve	NONE	A17	Reserve	NONE
A8	Display CAN LOW	BK/BU	A18	Display CAN HIGH	YE/BU
A9	Diagnostic CAN LOW	BK/RD	**A19**	Diagnostic CAN HIGH	YE/RD
A10	Reserve (Komfort 2)	NONE	A20	Reserve (Komfort 2)	NONE

Gateway Observations

The gateway seems to be the same as an Audi / VW PQ35 platform CAN gateway and the driver side door module doesn't even have a Porsche part number. (F005S00146)

Driver Side Door Module

The driver side door module is the connection between the 'memory buttons and the seat memory module. It also controls the mirrors which also have 'remembered' positions. This seems to be the missing element, and my hope / assumption is that if it (or messages injected from it) would bring the seat to life.



Door Control Unit – Left
C5 – CAN High Comfort
C6 – CAN Low Comfort
A27 – Memory Switch Position 1/2
A28 – Memory Switch Memory
C8, C19 - Term 31 (Chassis Gnd)
A22 – Term 58 (Accessory Power)

Memory Switch
A1 - Memory

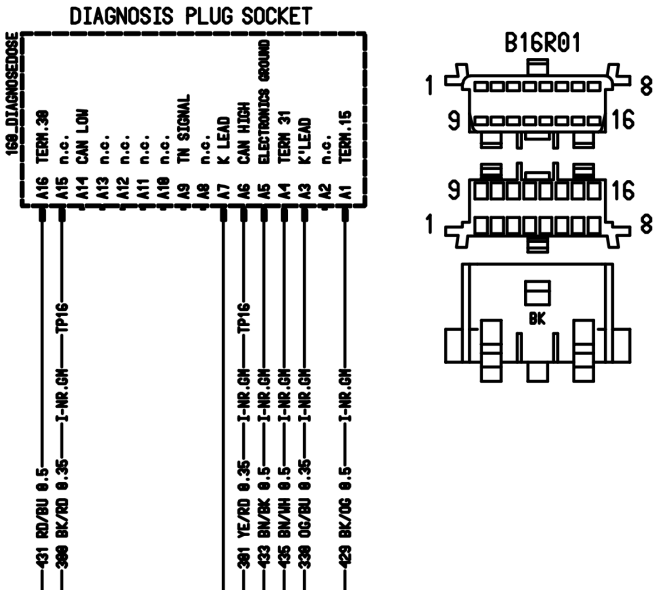
A2 – Term 31 (Chassis Gnd)
A3 – Position ID
A1 – Term 58 (Accessory Power)

(Plug A: 32 pins) - Misc
(Plug B: 16 pins) - Mirrors
(Plug C: 20 pins) - Motors

987 Memory Seat Buttons

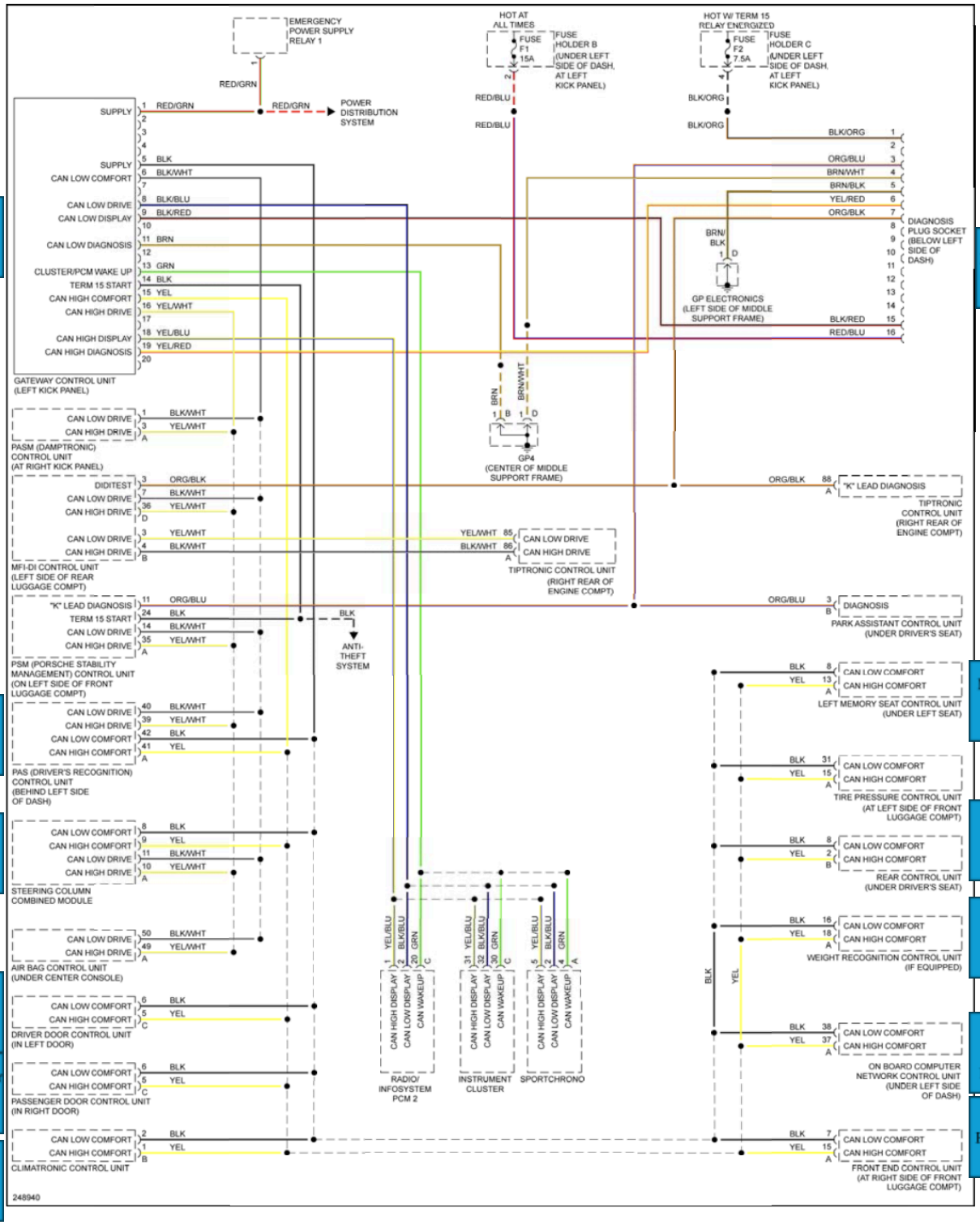
The memory buttons do not seem to be CAN based communication. The box does seem oddly large however the bus is not identified in the wiring diagrams.

Diagnosis OBD II Port



PIN	Description		PIN	Description
A1	Term 15 (Ignition +12V)		**A9**	TN SIGNAL (Engine RPM Pulse)
A2	n.c.		**A10**	n.c.
A3	K'LEAD (abs, airbag, psm, etc.)		**A11**	n.c.
A4	TERM 31 (Chassis Gnd)		**A12**	n.c.
A5	Electronics Ground		**A13**	n.c.
A6	CAN High (J-2234)		**A14**	CAN Low (J-2234)
A7	K LEAD (ISO 914-2) (alarm->DME)		**A15**	n.c.
A8	n.c.		**A16**	TERM 30 (Battery +12)

987 Comfort CAN Network



Gateway CU
997.610.107.04

PAS CU
(Porsche Access System)
997.618.171.17

Steering Column CU

Driver Door CU
F005S00329

Passenger Door CU
997.610.102.06

Climatronic CU

Diagnosis Port (ODBII Plug)

Memory Seat CU
997.618.537.07

Rear CU

Weight Recognition CU

On Board Computer Network CU

Front End CU

987 Comfort CAN Traffic

Seat Control Unit with Gateway Connected (no Ignition)

- Powered up with +12V and GND to both motor and electronics terminals
- Sources ~0.20mA of power for about 10 seconds then powers down to ~0.00mA

Seat Control Unit

- 0x406
    - Start-up: 06 02 04
    - Ready: 0B 01 00 00 00 40
    - Request Sleep: 0B 11
    - Shutdown: 0B 31
- 0x40B
    - Start-up: 0B 02 04 02
    - Ready: 06 01
    - Shutdown: 06 11

Gateway (No Ignition = no +12 on Terminal 15 input)

0x224
- Always: 00 00 7F 00 00 7F 00 1F

0x235
- Always: FF 7F 00 00 78 7F 00 1F

0x525
- Always: FF FF 0F 00 00 00 00 01

0x62F
- Always: 7D DF 9C 02 45 00 80 06

0x706
- Always: 22 50 19 04 06 00 00 00

0x70B
- Always: 16 50 16 F1 06 50 00 18

- **Next step will be to capture the CAN Data from a working car. Need to find the wake-up sequence for the Seat Control Unit.**

Major Breakthrough!

When I repeatedly send the following CAN Message the seat 'click's and comes to life.

CAN ID: 0x165
Payload Data: 0x47

But it must hear this frequently or it goes back to sleep.

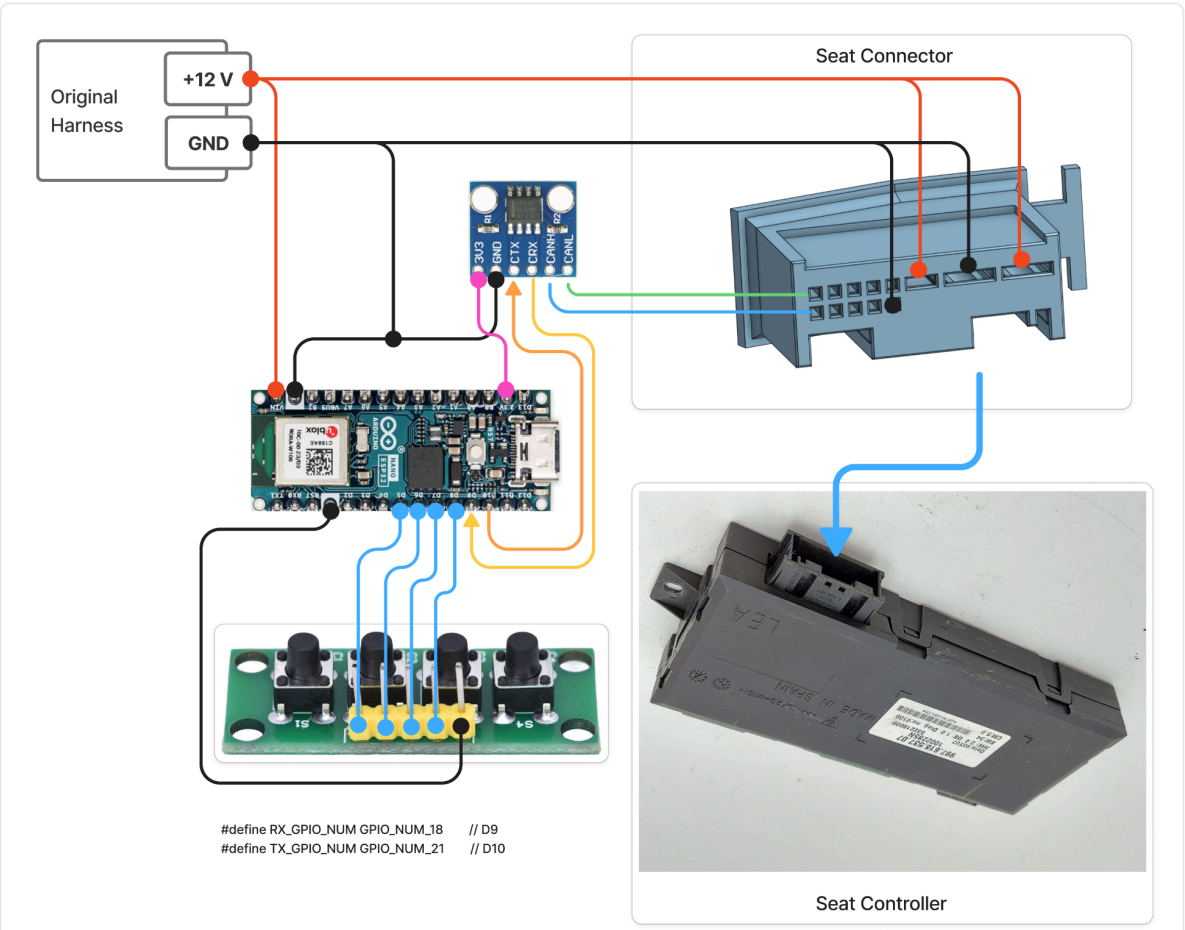
Go back to sleep time is right around 70 seconds.

Only drawback is that after about 3 seconds of seat movement, the seat stops and you have to unpress/repress the motion button you were just pressing. It's annoying but 100% usable. Planning to get back into figuring that part out!

Seat Controller Circuit Diagram
(Built and working)

Note: Buttons are not necessary. I added them for future functionality

Section 1



1. **Microcontroller: Arduino Nano ESP32**

   Important for 2 reasons:
   1. Supports 12 v natively so no need for power conversion circuits
   2. Has the CAN phy built in (tx on one line, rx on the other)

   Amazon Link:
   https://www.amazon.com/dp/B0C947C9Q...r_cso_cp_apin_dp_TZ03P6J3Z6SSETMJYXQX&csmig=1

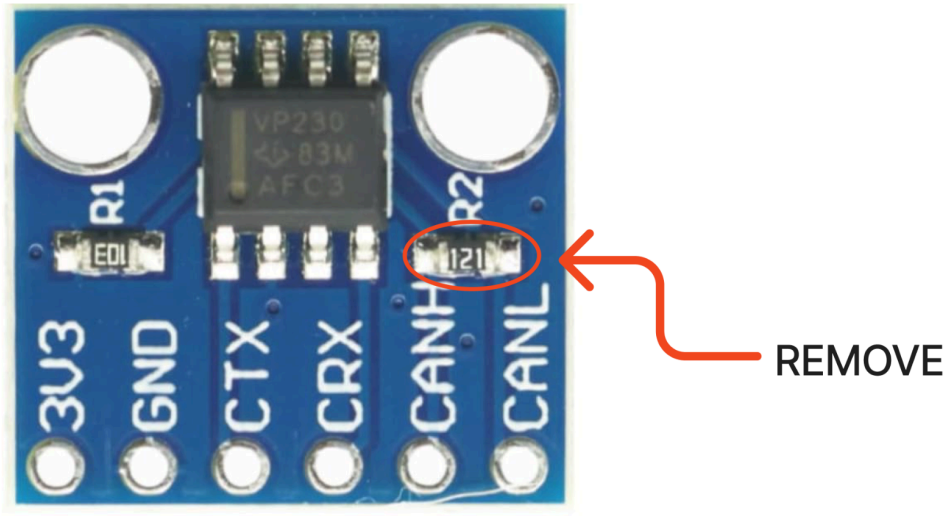
2. **CAN Transceiver: TI SN65HVD230**

   It needs to wire to a CAN Driver. The TI part is 3.3V and works either way the Arduino Nano.
   It's job is to take the tx/rx lines and convert them to a differential pair needed for the CAN bus. (CAN H / CAN L)

   I used an eval board from Amazon. But you need to make one modification: remove the 120 ohm load resistor between CAN H/L on that board.

Modify CAN Board

Remove Resistor



The TI part can be powered off the Arduino board's 3.3V out.

Amazon Link:
https://www.amazon.com/dp/B0DXDR1ZJ...r_cso_cp_apin_dp_EQWCXK3KKRYP36N0K95A&csmig=1

3. **Optional Buttons: 4 Key Keyboard Module 4 Keys**... https://www.amazon.com/dp/B09WB1VD97

3D Model of the Connector

[Onshape Link to the Seat Connector]

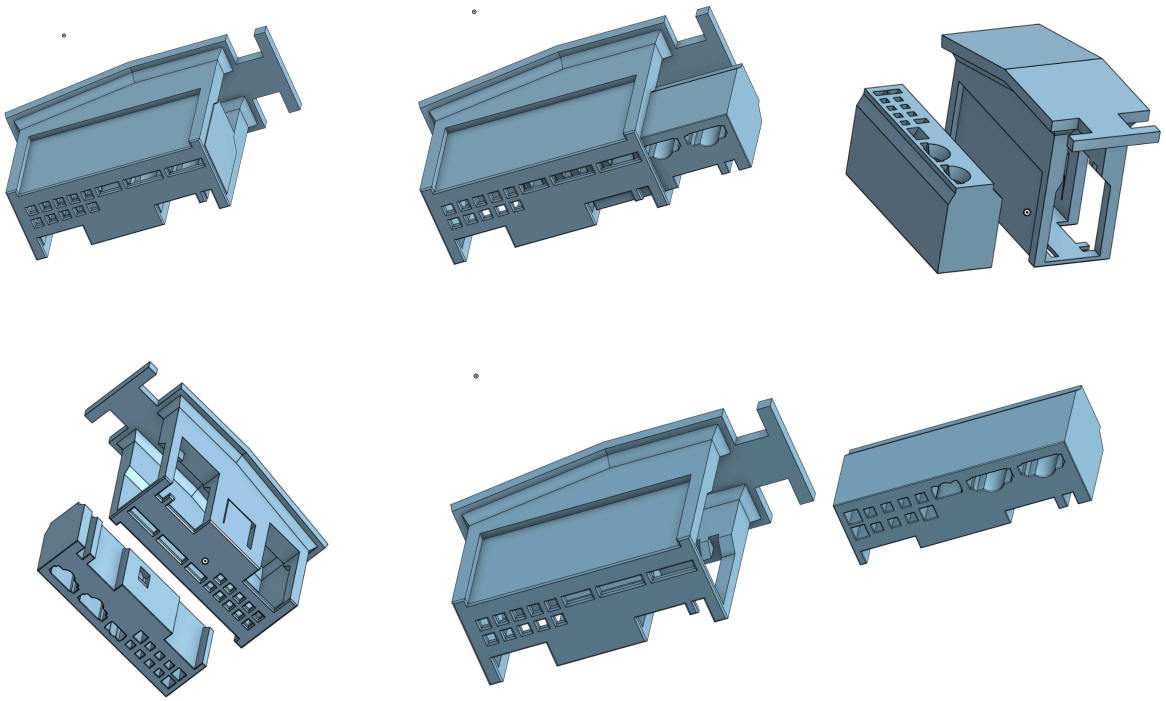
Connector is designed in 2 pieces.
Once wires are inserted from the bottom of the middle piece, they are mated allowing secure insertion and removal. A tab on top provides a place to attach a zip tie for strain relief.

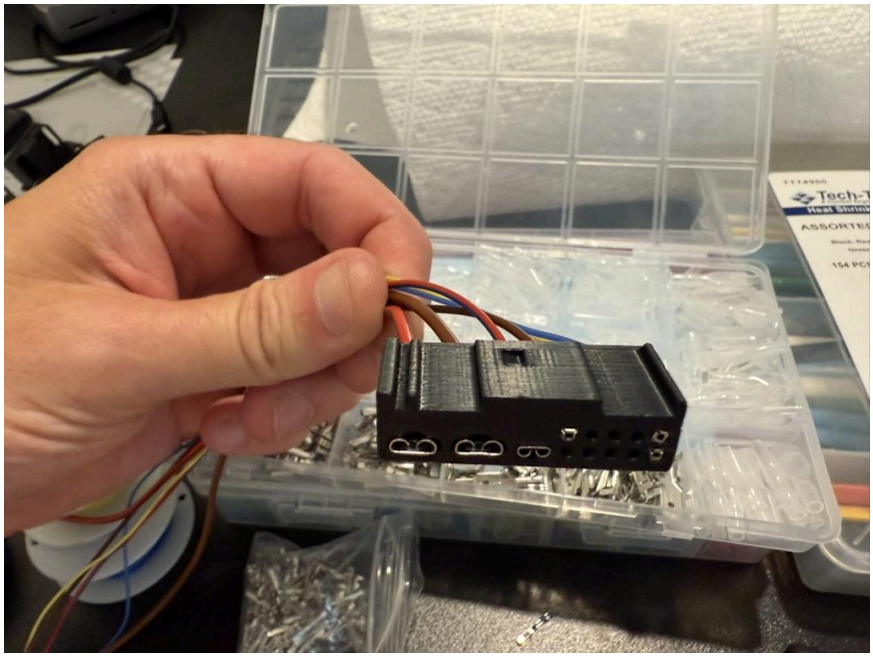
Required pieces:

- 4 Spade connectors (3 wide, 1 narrow)
- 3 socket female connectors

Where to get the parts?

- Spade connectors: [Amazon Link]
- Socket female connectors: [Amazon Link] (note that I did not buy these, but they look similar)
- Crimp tool (highly recommended): [Amazon Link] (have not used this, mine is similar)





Black inner connector. Insert wire ends first from the bottom and then press the female connectors into the holes. This inner body then slides into the outer shell.

Blue Outer Shell (have since printed a black one!)
I did end up using the upper yellow connector for the heated seat connections.

Arduino Code for Sending Seat Commands over CAN

This sends some extra commands that I was experimenting with…. The only issue is when the seat moves it stops after 3 seconds and you have to re-engage the button. I stopped working on it when I got to this point and this is my last working code drop…. I plan to get back into it and figure out how to operate continuously without the 3 sec timeout.

So the theory of operation is:

1. Press one of the 4 buttons
2. Microcontroller transmits CAN for 10 minutes and you can operate the seat
3. Then it goes back to sleep (and seat locks out after 70 seconds of no CAN messages)
4. Waits for button press to go back to step 1

The overall goal was to minimize possible battery drain by allowing the seat to lock out when not in use.
All would be solved if I simply wired the microcontroller into the ignition power but I haven't gotten around to doing that yet.

The other goal was to turn the buttons into memory seat buttons, but that seems like a far off dream at this point.

```
/* ====================================================================
 Arduino Nano ESP32-S3  –  Buttons + CAN/TWAI demo  (mixed rates)
 --------------------------------------------------------------------
 Buttons (active-low):  D5→GPIO8, D6→GPIO9, D7→GPIO10, D8→GPIO17
 CAN/TWAI pins:         D9→GPIO18 (RX), D10→GPIO21 (TX)
 Frame rates:
    ID 0x165 → 1 s
    ID 0x325 → 400 ms
    ID 0x260 → 400 ms
 ==================================================================== */

#include <Arduino.h>
#include "driver/twai.h"
#include "esp_wifi.h"
#include "esp_bt.h"
#include "WiFi.h"

/* ---------- pins ------------------------------------------------- */
#define RX_GPIO_NUM GPIO_NUM_18       // D9
#define TX_GPIO_NUM GPIO_NUM_21       // D10

const uint8_t BUTTON_PINS[4] = { D5, D6, D7, D8 };
const char*   BUTTON_TAG [4] = { "Button 1", "Button 2", "Button 3", "Button 4" };

/* ---------- timing ----------------------------------------------- */
const unsigned long BTN_SAMPLE_MS   = 10UL;
const unsigned long HEALTH_MS       = 1000UL;
```

```
const unsigned long ACTIVE_TIMEOUT  = 60000UL * 5UL; // 5 minutes          // Debug, use 10000UL;
const unsigned long BUS_RECOVER_MS  = 2000UL;

/* one interval per frame (ms) */
const unsigned long FRAME_INTERVAL[3] = { 500UL, 250UL, 250UL };

/* ---------- globals ---------------------------------------------------- */
bool lastLevel[4]          = { HIGH, HIGH, HIGH, HIGH };
unsigned long t_lastSample   = 0;
unsigned long t_lastHealth   = 0;
unsigned long t_activeStart  = 0;
unsigned long t_lastSend[3]  = { 0, 0, 0 };

/*  ▼▼▼  CHANGE 1 — start in sleep/paused state  ▼▼▼  */
bool txEnabled    = false;          // was true
/*  ▲▲▲----------------------------------------------▲▲▲  */

bool busIsHealthy = true;

/* ---------- CAN frames ------------------------------------------------- */
#define NUM_FRAMES 3
twai_message_t frame[NUM_FRAMES];

/* ----------------------------------------------------------------------- */
void disableWiFiAndBT() {
 WiFi.disconnect(true); WiFi.mode(WIFI_OFF);
 esp_wifi_stop();      esp_wifi_deinit();
 if (esp_bt_controller_get_status() == ESP_BT_CONTROLLER_STATUS_ENABLED)
    esp_bt_controller_disable();
 if (esp_bt_controller_get_status() != ESP_BT_CONTROLLER_STATUS_IDLE)
    esp_bt_controller_deinit();
 btStop();
}

void prepareFrames() {
 memset(&frame[0], 0, sizeof(twai_message_t));
 frame[0].identifier = 0x165; frame[0].data_length_code = 1;
 frame[0].data[0]    = 0x47;

 memset(&frame[1], 0, sizeof(twai_message_t));
```

```
frame[1].identifier = 0x325; frame[1].data_length_code = 8;
uint8_t d1[8] = {0xFF,0x7F,0x00,0x00,0x78,0x2E,0x00,0x0F};
memcpy(frame[1].data, d1, 8);

memset(&frame[2], 0, sizeof(twai_message_t));
frame[2].identifier = 0x260; frame[2].data_length_code = 8;
uint8_t d2[8] = {0x00,0x03,0x00,0x00,0x12,0x00,0x00,0x00};
memcpy(frame[2].data, d2, 8);
}

/* ---------------------------------------------------------------------- */
void recoverFromBusOff() {
Serial.println("BUS-OFF – cooling 2 s");
twai_stop();
delay(BUS_RECOVER_MS);
twai_driver_uninstall();
ESP.restart();                          // guaranteed recovery
}

bool checkBus() {
twai_status_info_t s; twai_get_status_info(&s);
if (s.state == TWAI_STATE_BUS_OFF) recoverFromBusOff();
return s.state == TWAI_STATE_RUNNING;
}

/* ============================  setup  ================================ */
void setup() {
Serial.begin(115200);
delay(50);
Serial.println("\n=== Nano ESP32 | Mixed-rate CAN demo ===");

disableWiFiAndBT();

twai_general_config_t g =
  TWAI_GENERAL_CONFIG_DEFAULT(TX_GPIO_NUM, RX_GPIO_NUM, TWAI_MODE_NORMAL);
twai_timing_config_t  t = TWAI_TIMING_CONFIG_100KBITS();
twai_filter_config_t  f = TWAI_FILTER_CONFIG_ACCEPT_ALL();

/*  ▼▼▼  CHANGE 1 — only install the driver here; don't start TWAI yet  ▼▼▼  */
if (twai_driver_install(&g,&t,&f) != ESP_OK) {
```

```
  Serial.println("TWAI driver install failed"); while (true) {}
}
/*  ▲▲▲----------------------------------------------------------▲▲▲  */

prepareFrames();
for (uint8_t p: BUTTON_PINS) pinMode(p, INPUT_PULLUP);

unsigned long now = millis();
t_activeStart = now;
for (int i=0;i<NUM_FRAMES;++i) t_lastSend[i] = now;   // start timers
}

/* ============================  loop  ================================= */
void loop() {
unsigned long now = millis();

/* ---- 1. sample buttons every 10 ms ------------------------------- */
if (now - t_lastSample >= BTN_SAMPLE_MS) {
  t_lastSample = now;
  for (uint8_t i=0;i<4;++i) {
    bool level = digitalRead(BUTTON_PINS[i]);
    if (level != lastLevel[i]) {                // edge
      Serial.printf("%s %s\n", BUTTON_TAG[i],
                    level==LOW ? "PRESSED" : "RELEASED");
      if (level==LOW) {                         // on press
        if (!txEnabled) {
          Serial.println("Resuming CAN-TX ...");
          if (twai_start()==ESP_OK) {           //  ← controller starts here
            txEnabled   = true;
            busIsHealthy = true;
          } else {
            Serial.println("twai_start() failed");
          }
        }
        t_activeStart = now;                    // reset 60 s timer
      }
      lastLevel[i] = level;
    }
  }
}
```

```
/* ---- 2. health check once per second ------------------------------ */
if (txEnabled && now - t_lastHealth >= HEALTH_MS) {
  busIsHealthy   = checkBus();
  t_lastHealth   = now;
  Serial.println(busIsHealthy ? "Bus OK" : "Bus waiting …");
}

/* ---- 3. transmit frames at individual intervals ------------------- */
if (txEnabled && busIsHealthy) {
  for (uint8_t i=0;i<NUM_FRAMES;++i) {
    if (now - t_lastSend[i] >= FRAME_INTERVAL[i]) {
      if (twai_transmit(&frame[i], pdMS_TO_TICKS(1000)) == ESP_OK) {
        Serial.printf("Sent 0x%03X  DLC=%u\n",
                      frame[i].identifier, frame[i].data_length_code);
      } else {
        Serial.println("TX error – pausing until bus healthy");
        busIsHealthy = false;
        break;                          // exit for-loop
      }
      t_lastSend[i] = now;
    }
  }
}

/* ---- 4. auto-timeout after 60 s ---------------------------------- */
if (txEnabled && now - t_activeStart >= ACTIVE_TIMEOUT) {
  Serial.println("60 s timeout — CAN-TX stopped");
  twai_stop();
  txEnabled = false;
}

delay(2);                               // keep loop responsive
}
```

How to Flash the Arduino Nano ESP32

1) Install the Arduino Software

1. Download and install the latest Arduino IDE.
2. In the IDE, go to Tools → Board → Board Manager and install Arduino ESP32 Boards.
3. From Tools → Board, select Arduino Nano ESP32.
4. (Optional but recommended) Install the ESP32 libraries from Espressif via Library Manager if your sketch uses them.
5. Connect your Nano ESP32 to your computer via USB-C and, in Tools → Port, choose the Nano's USB-C serial port. The board can be programmed directly over this port.

---

2) Load the Sketch

1. Create a New Sketch in the Arduino IDE.
2. Paste in the Nano ESP32 program from above.
3. Click Verify (✓) to compile, then click Upload (→) to flash it to the board.
4. Open the Serial Monitor (115200 baud) to confirm the board is running your code.