

```

1  //-----
2  // Code inspired by - Model Railroading with Arduino
3  //   - NCE Auxiliary Input Unit (AIU) Example - (c) 2019 Alex Shepherd
4  // Code modified by Terry Chamberlain to run without use of NceCabBus library - March 2021
5  //   - Corrected to run with Power Cab and JMRI Decoder Pro - February 2026
6  // Allows setting of Cab Bus Address via USB Serial connection when pin 20 (D21, A3) held LOW
7  // .. Cab Bus Address value set on Inputs 1-6 and input when pin 18 (D18, A0) set LOW
8  // .. Set Address is held in EEPROM @ address 256 (requires EEPROM library)
9  // Also allows Inputs 10-14 to be changed to outputs which reflect the states of either
10 // .. Inputs 1-5 or Inputs 5-9 - Output Mode is set by inputting a Cab Bus Address with
11 // .. either Input 8 set LOW to monitor Inputs 1-5, or Input 7 set LOW to monitor Inputs 5-9
12 // .. Output Mode is held in EEPROM @ address 257, Mode active when EEPROM @ 257 = 0x05 to
13 // .. monitor Inputs 1-5 or EEPROM @ 257 = 0x0A to monitor Inputs 5-9
14 // .. Each of Outputs 10-14 are set HIGH when the corresponding Inputs 1-5 or 5-9 are set LOW
15 //-----
16
17 #include <EEPROM.h>
18 #include <Bounce2.h>
19
20 // Change the #define below to match the Serial port you're using for RS485
21 #define RS485Serial Serial1
22
23 // Change the #define below to match the RS485 Chip TX Enable pin
24 // #define RS485_TX_ENABLE_PIN 2
25 byte RS485_TX_ENABLE_PIN = 2;
26
27 // Set the default AIU Cab Bus Address - altered to value in EEPROM read in void(setup)
28 byte CAB_BUS_ADDRESS = 5;
29 byte check_ping = 0x85; // Set value to check for ping to this LIM
30
31 #define DebugMonitor Serial
32 // Uncomment the line below to enable Debug Output
33 byte DEBUG = 1;
34
35 // Set the Number of AIU Inputs to be scanned
36 byte NUM_AIU_INPUTS = 14;
37
38 // The Array below maps Arduino Pins to AIU Inputs, change as required
39 // .. AIU Input Numbers 1 2 3 4 5 6 7 8 9 10 11 12 13 14
40 byte aiuInputPins[] = {3,4,5,6,7,8,9,20,19,18,15,14,16,10};
41
42 // Change the #define below to set the Number of Debounce milliseconds for the AIU Inputs
43 // #define DEBOUNCE_MS 20
44
45 Bounce * debAIUins = new Bounce[NUM_AIU_INPUTS];
46
47 // EEPROM used to hold current AIU Cab Bus Address and QSDM Mode
48 int eepm_addr = 256; // Use location above first page (0 - 255)
49 byte eepm_data = 0;
50 byte numswpins = 8;
51 byte swpins[] = {3,4,5,6,7,8,9,20}; // Digital pins for Cab Bus Setting switches (D20 = A2)
52 byte prog_actv = 0;
53 byte prog_redy = 0;

```

```

54 byte prog_done = 0;
55 byte outp_mode = 0;
56 byte flash = 0;
57 const byte adr_prog = 21; // Prepare to set AIU Cab Bus Address when pin D21 (A3) pulled low
58 const byte prog_led = 19; // D19 (A1) drives LED to show programming AIU Cab Bus Address
59 const byte prog_set = 18; // Set AIU Cab Bus Address when pin D18 (A0) pulled low
60 int i; // Loop index
61
62 const char* splashMsg = "Quad Servo Decoder Monitor - Version 2.8";
63 // AIU Input Index - used to update one aiuInput per loop
64 byte aiuInputIndex = 0;
65
66 byte in_Cmnd = 0; // Serial command in
67 byte in_Data = 0; // Count of serial data bytes in (from Command Station)
68 byte out_Buf[] = {0x3F, 0x3F}; // Serial data out - state of AIU pins 1-7 and 8-14
69 const byte typ_AIU = 0x64; // AIU Cab Type = 'd'
70 byte pingAIU = 0; // Flag set if this AIU just pinged - reset if ping not for this AIU
71
72 void setup() {
73     uint32_t startMillis = millis();
74
75     DebugMonitor.begin(115200);
76     while (!DebugMonitor && ((millis() - startMillis) < 3000)); // wait for serial port to connect. Needed for native USB
77
78     DebugMonitor.println();
79     DebugMonitor.println(splashMsg);
80
81     pinMode(adr_prog, INPUT_PULLUP); // Set AIU Cab Bus Address when pin D21 pulled low
82     eepm_data = EEPROM.read(256);
83     if (eepm_data > 63 || eepm_data < 2){
84         eepm_data = 5; // Set default AIU Cab Address = 5
85     }
86     CAB_BUS_ADDRESS = eepm_data;
87     check_ping = 0x80 | (CAB_BUS_ADDRESS & 0x3F);
88     pingAIU = 0;
89     in_Data = 0;
90     prog_actv = 0;
91     aiuInputIndex = 0;
92     out_Buf[0] = 0x3F;
93     out_Buf[1] = 0x3F;
94
95     eepm_data = EEPROM.read(257);
96     if (eepm_data == 0x05){
97         outp_mode = 0x05; // Set Output Mode active for Inputs 1-5
98         NUM_AIU_INPUTS = 9;
99     }
100     else if (eepm_data == 0x0A) {
101         outp_mode = 0x0A; // Set Output Mode active for Inputs 5-9
102         NUM_AIU_INPUTS = 9;
103     }
104     else {
105         outp_mode = 0; // Set Output Mode inactive
106         NUM_AIU_INPUTS = 14;

```

```

107 }
108
109 pinMode(RS485_TX_ENABLE_PIN, OUTPUT);
110 digitalWrite(RS485_TX_ENABLE_PIN, LOW);
111 RS485Serial.begin(9600, SERIAL_8N2);
112
113 for(uint8_t i = 0; i < NUM_AIU_INPUTS; i++) {
114   debAIUins[i].attach(aiuInputPins[i], INPUT_PULLUP); // Setup the bounce instance and mode for each AIU Input pin
115   debAIUins[i].interval(20); // .. with a debounce period of 20msec
116   if (i < 7) {
117     if (digitalRead(aiuInputPins[i]) == HIGH) {
118       bitSet(out_Buf[0], i); // Initialise output status to reflect state of AIU Input pins
119     }
120     else {
121       bitClear(out_Buf[0], i);
122     }
123   }
124   else {
125     if (digitalRead(aiuInputPins[i]) == HIGH) {
126       bitSet(out_Buf[1], i - 7);
127     }
128     else {
129       bitClear(out_Buf[1], i - 7);
130     }
131   }
132 }
133 if (outp_mode == 0x05 || outp_mode == 0x0A) {
134   for(uint8_t i = 9; i < 14; i++) {
135     pinMode(aiuInputPins[i], OUTPUT); // Change Input Pins 10-14 to Output
136     digitalWrite(aiuInputPins[i], LOW); // Switch any attached LED off
137     bitSet(out_Buf[1], i - 7); // Report state of these Input Pins as HIGH (not active)
138   }
139 }
140 }
141
142 void loop() {
143   // ===== Check for Address Programming Link fitted =====
144   if (digitalRead(adr_prog) == LOW && prog_actv == 0) {
145     delay(200);
146     if (digitalRead(adr_prog) == LOW) {
147       // Address Programming Link fitted - start data entry
148       //*****
149       DebugMonitor.println(splashMsg);
150       DebugMonitor.println("Address Programming Start");
151       DebugMonitor.print("Current Address : ");
152       DebugMonitor.println(CAB_BUS_ADDRESS, DEC);
153       if (outp_mode == 0x05) {
154         DebugMonitor.println("Output Mode : Monitoring Inputs 1-5");
155       }
156       else if (outp_mode == 0x0A) {
157         DebugMonitor.println("Output Mode : Monitoring Inputs 5-9");
158       }
159       else {

```

```

160     DebugMonitor.println("Output Mode      : Inactive");
161 }
162 //*****
163 prog_actv = 1;           // Set programming active flag
164 prog_redy = 1;           // Set ready for input flag
165 pinMode(prog_led, OUTPUT); // Change pin D19 to Output
166 digitalWrite(prog_led, HIGH); // Switch programming LED on
167 pinMode(prog_set, INPUT_PULLUP); // Ensure pin D18 (Programming pushbutton) is Input
168 // Ready to set AIU Cab Bus Address
169 while (prog_actv == 1) {
170     eepm_data = 0;
171     for(i = 7; i > -1; i--) {
172         // Read address switches 8 > 1 in sequence
173         eepm_data = eepm_data << 1; // Shift programming switch data left (x2)
174         if (digitalRead(swpins[i]) == LOW) {
175             eepm_data = eepm_data + 1; // Add 1 to Address if switch[i] is On
176         }
177     }
178     if (digitalRead(prog_set) == LOW && prog_done == 0) {
179         delay(200);
180         if (digitalRead(prog_set) == LOW) {
181             // Pushbutton pressed - address programming done
182             //*****
183             DebugMonitor.println("Address Entered");
184             //*****
185             prog_done = 1; // Set data accepted flag
186             prog_redy = 0; // Clear ready for input flag
187             digitalWrite(prog_led, LOW); // Switch programming LED off
188             pinMode(prog_led, INPUT_PULLUP); // Change pin D19 back to Input
189             if (digitalRead(adr_prog) == LOW) {
190                 // Address Programming Link still fitted
191                 if ((eepm_data & 0x3F) > 2) {
192                     EEPROM.write(256,(eepm_data & 0x3F)); // Save entered Cab Bus Address if valid (>02)
193                     CAB_BUS_ADDRESS = EEPROM.read(256);
194                     //*****
195                     DebugMonitor.print("Address Stored : ");
196                     DebugMonitor.println(CAB_BUS_ADDRESS, DEC) ;
197                     //*****
198                 }
199                 if ((eepm_data & 0x80) != 0) {
200                     EEPROM.write(257,0x05); // Set Output Mode = Monitor Inputs 1-5
201                     outp_mode = EEPROM.read(257);
202                     //*****
203                     DebugMonitor.println("Output Mode      : Monitoring Inputs 1-5");
204                     //*****
205                 }
206                 else if ((eepm_data & 0x40) != 0) {
207                     EEPROM.write(257,0x0A); // Set Output Mode = Monitor Inputs 5-9
208                     outp_mode = EEPROM.read(257);
209                     //*****
210                     DebugMonitor.println("Output Mode      : Monitoring Inputs 5-9");
211                     //*****
212                 }

```

```

213     else if ((eepm_data & 0xC0) == 0) {
214         EEPROM.write(257,0); // Set Output Mode = Inactive
215         outp_mode = EEPROM.read(257);
216         //*****
217         DebugMonitor.println("Output Mode      : Inactive");
218         //*****
219     }
220 }
221 }
222 }
223 if (digitalRead(adr_prog) == HIGH && prog_done == 1) {
224     delay(200);
225     if (digitalRead(adr_prog) == HIGH) {
226         // Programming Link removed after successful programming
227         prog_done = 0; // Clear programmed flag
228         prog_redy = 0; // Clear ready for input flag
229         prog_actv = 0; // Disable Address Programming
230         //*****
231         DebugMonitor.println("Address Programming Completed") ;
232         //*****
233     }
234 }
235 if (digitalRead(adr_prog) == HIGH && prog_redy == 1) {
236     delay(200);
237     if (digitalRead(adr_prog) == HIGH) {
238         // Programming Link removed without programming
239         //*****
240         DebugMonitor.println("Address Programming Abandoned") ;
241         //*****
242         digitalWrite(prog_led, LOW); // Switch programming LED off
243         pinMode(prog_led, INPUT_PULLUP); // Change pin D19 back to Input
244         prog_done = 0; // Clear programmed flag
245         prog_redy = 0; // Clear ready for input flag
246         prog_actv = 0; // Disable Address Programming
247     }
248 }
249 } // End while(prog_actv)
250 } // End Confirm(adr_prog)
251 } // End Read(adr_prog)
252
253 // ===== Otherwise proceed with normal AIU operation =====
254 else if (prog_actv == 0)
255 {
256     // Check first if a command has been received
257     if (RS485Serial.available())
258     {
259         // Read incoming command stream from Command Station
260         in_Cmnd = RS485Serial.read();
261         if (in_Data != 0)
262         {
263             // Data command has been received from Command Station
264             for (i = 1; i < in_Data; i++)
265             {

```

```

266     while (RS485Serial.available()); // Check that data is in buffer
267     in_Cmnd = RS485Serial.read();    // Read and ignore incoming data bytes
268 }
269 in_Data = 0; // Clear data byte count
270 }
271 if ((in_Cmnd & 0xF0) == 0xC0 || (in_Cmnd & 0xF0) == 0xD0)
272 {
273     // Input is Command from Command Station
274     if (in_Cmnd < 0xC9)
275     {
276         in_Data = 8; // Expect 8 data bytes to follow
277     }
278     else if ((in_Cmnd < 0xCB) || (in_Cmnd == 0xCC))
279     {
280         in_Data = 1; // Expect 1 data byte to follow
281     }
282     else if (in_Cmnd == 0xCB)
283     {
284         in_Data = 9; // Expect 9 data bytes to follow
285     }
286     else if (in_Cmnd == 0xD8)
287     {
288         in_Data = 2; // Expect 2 data bytes to follow
289     }
290     else if (in_Cmnd == 0xDB)
291     {
292         in_Data = 4; // Expect 4 data bytes to follow
293     }
294 }
295 if((in_Cmnd & 0xC0) == 0x80)
296 {
297     if (DEBUG == 1)
298     {
299         DebugMonitor.println(); // Ping received
300         DebugMonitor.print("P:");
301         DebugMonitor.print(in_Cmnd, HEX);
302         DebugMonitor.print(' ');
303     }
304     else
305     {
306         delayMicroseconds(400);
307     }
308     pingAIU = 0; // Clear ping flag
309 }
310 if (in_Cmnd == 0xD2 && pingAIU == 1)
311     // AIU must wait at least 100usec and not more than 780usec before responding to make sure
312     // the Command Station (RS485 Master) has disabled transmission and is ready for a response
313 {
314     delayMicroseconds(400); // Pause before sending response
315     digitalWrite(RS485_TX_ENABLE_PIN, HIGH); // Enable transmit response on RS485 port
316     RS485Serial.write(typ_AIU); // Output AIU Cab Type 'd'
317     RS485Serial.flush(); // Wait until data sent
318     delayMicroseconds(100); // Pause after sending response

```

```

319     digitalWrite(RS485_TX_ENABLE_PIN, LOW); // .. then disable transmit / enable receive
320     if (DEBUG == 1)
321     {
322         DebugMonitor.println();
323         DebugMonitor.print("T:");
324         DebugMonitor.print(typ_AIU, HEX);
325         DebugMonitor.print(' ');
326     }
327     else
328     {
329         delayMicroseconds(400);
330     }
331     pingAIU = 0; // Clear ping flag
332 }
333 if (in_Cmnd == check_ping)
334 {
335     pingAIU = 1; // Ping is for this AIU
336     delayMicroseconds(400);
337     digitalWrite(RS485_TX_ENABLE_PIN, HIGH); // Enable transmit response on RS485 port
338     RS485Serial.write(out_Buf[0]); // Output AIU status - Inputs 1-7
339     delayMicroseconds(200);
340     RS485Serial.write(out_Buf[1]); // Output AIU status - Inputs 8-14
341     RS485Serial.flush(); // Wait until data sent
342     delayMicroseconds(100); // Pause before sending response
343     digitalWrite(RS485_TX_ENABLE_PIN, LOW); // .. then disable transmit / enable receive
344     if (DEBUG == 1)
345     {
346         DebugMonitor.println();
347         DebugMonitor.print("S:");
348         DebugMonitor.print(out_Buf[0], HEX);
349         DebugMonitor.print(' ');
350         DebugMonitor.print(out_Buf[1], HEX);
351         DebugMonitor.print(' ');
352     }
353 }
354 else
355 {
356     if (DEBUG == 1)
357     {
358         DebugMonitor.println();
359         DebugMonitor.print("R:");
360         DebugMonitor.print(in_Cmnd, HEX);
361         DebugMonitor.print(' ');
362     }
363     else
364     {
365         delayMicroseconds(400);
366     }
367 }
368 }
369 else
370 {
371     // If no command pending, debounce a single AIU Input and update its status

```

```

372 if (debAIUins[aiuInputIndex].update()) // Check for a change in next AIU Input pin
373 {
374     DebugMonitor.print("Input Changed: Index: ");
375     DebugMonitor.print(aiuInputIndex);
376     if (aiuInputIndex < 7)
377     {
378         if (digitalRead(aiuInputPins[aiuInputIndex]) == HIGH)
379         {
380             bitSet(out_Buf[0],aiuInputIndex);
381             DebugMonitor.print(" - Set Hi");
382             DebugMonitor.print(" - Out-0 = ");
383             DebugMonitor.println(out_Buf[0], HEX);
384         }
385         else
386         {
387             bitClear(out_Buf[0],aiuInputIndex);
388             DebugMonitor.print(" - Set Lo");
389             DebugMonitor.print(" - Out-0 = ");
390             DebugMonitor.println(out_Buf[0], HEX);
391         }
392     }
393     else
394     {
395         if (digitalRead(aiuInputPins[aiuInputIndex]) == HIGH)
396         {
397             bitSet(out_Buf[1], (aiuInputIndex - 7));
398             DebugMonitor.print(" - Set Hi");
399             DebugMonitor.print(" - Out-1 = ");
400             DebugMonitor.println(out_Buf[1], HEX);
401         }
402         else
403         {
404             bitClear(out_Buf[1], (aiuInputIndex - 7));
405             DebugMonitor.print(" - Set Lo");
406             DebugMonitor.print(" - Out-1 = ");
407             DebugMonitor.println(out_Buf[1], HEX);
408         }
409     }
410 }
411 if (outp_mode == 0x05 && aiuInputIndex < 5) {
412     if (digitalRead(aiuInputPins[aiuInputIndex]) == HIGH)
413     {
414         digitalWrite(aiuInputPins[aiuInputIndex + 9], LOW);
415     }
416     else
417     {
418         digitalWrite(aiuInputPins[aiuInputIndex + 9], HIGH);
419     }
420 }
421 else if (outp_mode == 0x0A && aiuInputIndex > 3 && aiuInputIndex < 9) {
422     if (digitalRead(aiuInputPins[aiuInputIndex]) == HIGH)
423     {
424         digitalWrite(aiuInputPins[aiuInputIndex + 5], LOW);

```

```
425     }
426     else
427     {
428         digitalWrite(aiuInputPins[aiuInputIndex + 5], HIGH);
429     }
430 }
431 aiuInputIndex++;
432 if(aiuInputIndex >= NUM_AIU_INPUTS)
433     aiuInputIndex = 0;
434 }
435 }
436 delayMicroseconds(500);
437 } // End loop
```