

```

//-----
// Code inspired by - Model Railroading with Arduino
// - NCE Auxiliary Input Unit (AIU) Example - (c) 2019 Alex Shepherd
// Code modified by Terry Chamberlain to run without use of NceCabBus library - March 2021
// Allows setting of Cab Bus Address via USB Serial connection when pin 20 (D21, A3) held LOW
// .. Cab Bus Address value set on Inputs 1-6 and input when pin 18 (D18, A0) set LOW
// .. Set Address is held in EEPROM @ address 256 (requires EEPROM library)
// Also allows Inputs 10-14 to be changed to outputs which reflect the states of either
// .. Inputs 1-5 or Inputs 5-9 - Output Mode is set by inputting a Cab Bus Address with
// .. either Input 8 set LOW to monitor Inputs 1-5, or Input 7 set LOW to monitor Inputs 5-9
// .. Output Mode is held in EEPROM @ address 257, Mode active when EEPROM @ 257 = 0x05 to
// .. monitor Inputs 1-5 or EEPROM @ 257 = 0x0A to monitor Inputs 5-9
// .. Each of Outputs 10-14 are set HIGH when the corresponding Inputs 1-5 or 5-9 are set LOW
//-----

#include <EEPROM.h>
#include <Bounce2.h>

// Change the #define below to match the Serial port you're using for RS485
#define RS485Serial Serial1

// Change the #define below to match the RS485 Chip TX Enable pin
// #define RS485_TX_ENABLE_PIN 2
byte RS485_TX_ENABLE_PIN = 2;

// Set the default AIU Cab Bus Address - altered to value in EEPROM read in void(setup)
byte CAB_BUS_ADDRESS = 5;

#define DebugMonitor Serial
// Uncomment the line below to enable Debug Output
byte DEBUG = 1;

// Set the Number of AIU Inputs to be scanned
byte NUM_AIU_INPUTS = 14;

// The Array below maps Arduino Pins to AIU Inputs, change as required
// .. AIU Input Numbers 1 2 3 4 5 6 7 8 9 10 11 12 13 14
byte aiuInputPins[] = {3,4,5,6,7,8,9,20,19,18,15,14,16,10};

// Change the #define below to set the Number of Debounce milliseconds for the AIU Inputs
// #define DEBOUNCE_MS 20

Bounce * debAIUins = new Bounce[NUM_AIU_INPUTS];

// EEPROM used to hold current AIU Cab Bus Address and QSDM Mode
int eepm_addr = 256; // Use location above first page (0 - 255)
byte eepm_data = 0;
byte numswpins = 8;
byte swpins[] = {3,4,5,6,7,8,9,20}; // Digital pins for Cab Bus Setting switches (D20 = A2)
byte prog_actv = 0;
byte prog_redy = 0;
byte prog_done = 0;
byte outp_mode = 0;
byte flash = 0;
const byte adr_prog = 21; // Prepare to set AIU Cab Bus Address when pin D21 (A3) pulled low
const byte prog_led = 19; // D19 (A1) drives LED to show programming AIU Cab Bus Address
const byte prog_set = 18; // Set AIU Cab Bus Address when pin D18 (A0) pulled low
int i; // Loop index

const char* splashMsg = "Layout Input Monitor - Version 2.6";
// AIU Input Index - used to update one aiuInput per loop
byte aiuInputIndex = 0;

byte in_Cmnd = 0; // Serial command in
byte out_Buf[] = {0x7F, 0x7F}; // Serial data out - state of AIU pins 1-7 and 8-14
const byte typ_AIU = 0x64; // AIU Cab Type = 'd'
byte pingAIU = 0; // Flag set if this AIU just pinged - reset if ping not for this AIU

void setup() {
  uint32_t startMillis = millis();

```

```

DebugMonitor.begin(115200);
while (!DebugMonitor && ((millis() - startMillis) < 3000)); // wait for serial port to connect. Needed for
native USB

DebugMonitor.println();
DebugMonitor.println(splashMsg);

pinMode(adr_prog, INPUT_PULLUP); // Set AIU Cab Bus Address when pin D21 pulled low
eepm_data = EEPROM.read(256);
if (eepm_data > 63 || eepm_data < 2){
  eepm_data = 5; // Set default AIU Cab Address = 5
}
CAB_BUS_ADDRESS = eepm_data;
eepm_data = EEPROM.read(257);
if (eepm_data == 0x05){
  outp_mode = 0x05; // Set Output Mode active for Inputs 1-5
  NUM_AIU_INPUTS = 9;
}
else if (eepm_data == 0x0A) {
  outp_mode = 0x0A; // Set Output Mode active for Inputs 5-9
  NUM_AIU_INPUTS = 9;
}
else {
  outp_mode = 0; // Set Output Mode inactive
  NUM_AIU_INPUTS = 14;
}

pinMode(RS485_TX_ENABLE_PIN, OUTPUT);
digitalWrite(RS485_TX_ENABLE_PIN, LOW);
RS485Serial.begin(9600, SERIAL_8N2);

for(uint8_t i = 0; i < NUM_AIU_INPUTS; i++) {
  debAIUins[i].attach(aiuInputPins[i], INPUT_PULLUP); // Setup the bounce instance and mode for each AIU
Input pin
  debAIUins[i].interval(20); // .. with a debounce period of 20msec
  if (i < 7) {
    if (digitalRead(aiuInputPins[i]) == HIGH) {
      bitSet(out_Buf[0], i); // Initialise output status to reflect state of AIU Input pins
    }
    else {
      bitClear(out_Buf[0], i);
    }
  }
  else {
    if (digitalRead(aiuInputPins[i]) == HIGH) {
      bitSet(out_Buf[1], i - 7);
    }
    else {
      bitClear(out_Buf[1], i - 7);
    }
  }
}
if (outp_mode == 0x05 || outp_mode == 0x0A) {
  for(uint8_t i = 9; i < 14; i++) {
    pinMode(aiuInputPins[i], OUTPUT); // Change Input Pins 10-14 to Output
    digitalWrite(aiuInputPins[i], LOW); // Switch any attached LED off
    bitSet(out_Buf[1], i - 7); // Report state of these Input Pins as HIGH (not active)
  }
}
}

void loop() {
// ===== Check for Address Programming Link fitted =====
if (digitalRead(adr_prog) == LOW && prog_actv == 0) {
  delay(200);
  if (digitalRead(adr_prog) == LOW) {
    // Address Programming Link fitted - start data entry
    //*****
    DebugMonitor.println(splashMsg);
    DebugMonitor.println("Address Programming Start");
    DebugMonitor.print("Current Address : ");
  }
}
}

```



```

if (digitalRead(adr_prog) == HIGH && prog_done == 1) {
  delay(200);
  if (digitalRead(adr_prog) == HIGH) {
    // Programming Link removed after successful programming
    prog_done = 0; // Clear programmed flag
    prog_redy = 0; // Clear ready for input flag
    prog_actv = 0; // Disable Address Programming
    //*****
    DebugMonitor.println("Address Programming Completed");
    //*****
  }
}
if (digitalRead(adr_prog) == HIGH && prog_redy == 1) {
  delay(200);
  if (digitalRead(adr_prog) == HIGH) {
    // Programming Link removed without programming
    //*****
    DebugMonitor.println("Address Programming Abandoned");
    //*****
    digitalWrite(prog_led, LOW); // Switch programming LED off
    pinMode(prog_led, INPUT_PULLUP); // Change pin D19 back to Input
    prog_done = 0; // Clear programmed flag
    prog_redy = 0; // Clear ready for input flag
    prog_actv = 0; // Disable Address Programming
  }
}
} // End while(prog_actv)
} // End Confirm(adr_prog)
} // End Read(adr_prog)

// ===== Otherwise proceed with normal AIU operation =====
else if (prog_actv == 0)
{
  // Check first if a command has been received
  if (RS485Serial.available())
  {
    // Read incoming command stream from Command Station
    in_Cmnd = RS485Serial.read();

    if((in_Cmnd & 0xC0) == 0x80)
    {
      if (DEBUG == 1)
      {
        DebugMonitor.println();
      }
      else
      {
        delayMicroseconds(100);
      }
    }
  }
  if (DEBUG == 1)
  {
    DebugMonitor.print("R:");
    DebugMonitor.print(in_Cmnd, HEX);
    DebugMonitor.print(' ');
  }
  else
  {
    delayMicroseconds(400);
  }
}

if (in_Cmnd == 0xD2 && pingAIU == 1)
  // AIU must wait at least 100usec and not more than 780usec before responding to make sure
  // the Command Station (RS485 Master) has disabled transmission and is ready for a response
{
  delayMicroseconds(400); // Pause before sending response
  digitalWrite(RS485_TX_ENABLE_PIN, HIGH); // Enable transmit response on RS485 port
  RS485Serial.write(typ_AIU); // Output AIU Cab Type 'd'
  RS485Serial.flush(); // Wait until data sent
  delayMicroseconds(100); // Pause after sending response
  digitalWrite(RS485_TX_ENABLE_PIN, LOW); // .. then disable transmit / enable receive
}

```

```

if (DEBUG == 1)
{
  DebugMonitor.println();
  DebugMonitor.print("T:");
  DebugMonitor.print(typ_AIU, HEX);
  DebugMonitor.print(' ');
}
else
{
  delayMicroseconds(400);
}
pingAIU = 0; // Clear ping flag
}
if ((in_Cmnd & 0x7F) == CAB_BUS_ADDRESS)
{
  pingAIU = 1; // Ping is for this AIU
  delayMicroseconds(400);
  digitalWrite(RS485_TX_ENABLE_PIN, HIGH); // Enable transmit response on RS485 port
  RS485Serial.write(out_Buf[0]); // Output AIU status - Inputs 1-7
  delayMicroseconds(200);
  RS485Serial.write(out_Buf[1]); // Output AIU status - Inputs 8-14
  RS485Serial.flush(); // Wait until data sent
  delayMicroseconds(100); // Pause before sending response
  digitalWrite(RS485_TX_ENABLE_PIN, LOW); // .. then disable transmit / enable receive
  DebugMonitor.println();
  DebugMonitor.print("T:");
  DebugMonitor.print(out_Buf[0], HEX);
  DebugMonitor.print(' ');
  DebugMonitor.print(out_Buf[1], HEX);
  DebugMonitor.print(' ');
}
else
{
  pingAIU = 0; // Ping is not for this AIU
}
}
else
{
  // If no command pending, debounce a single AIU Input and update its status
  if(debAIUins[aiuInputIndex].update()) // Check for a change in next AIU Input pin
  {
    DebugMonitor.print("Input Changed: Index: ");
    DebugMonitor.print(aiuInputIndex);
    if (aiuInputIndex < 7)
    {
      if (digitalRead(aiuInputPins[aiuInputIndex]) == HIGH)
      {
        bitSet(out_Buf[0],aiuInputIndex);
        DebugMonitor.print(" - Set Hi");
        DebugMonitor.print(" - Out-0 = ");
        DebugMonitor.println(out_Buf[0], HEX);
      }
      else
      {
        bitClear(out_Buf[0],aiuInputIndex);
        DebugMonitor.print(" - Set Lo");
        DebugMonitor.print(" - Out-0 = ");
        DebugMonitor.println(out_Buf[0], HEX);
      }
    }
  }
  else
  {
    if (digitalRead(aiuInputPins[aiuInputIndex]) == HIGH)
    {
      bitSet(out_Buf[1], (aiuInputIndex - 7));
      DebugMonitor.print(" - Set Hi");
      DebugMonitor.print(" - Out-1 = ");
      DebugMonitor.println(out_Buf[1], HEX);
    }
    else
    {

```

```

        bitClear(out_Buf[1], (aiuInputIndex - 7));
        DebugMonitor.print(" - Set Lo");
        DebugMonitor.print(" - Out-1 = ");
        DebugMonitor.println(out_Buf[1], HEX);
    }
}
}
if (outp_mode == 0x05 && aiuInputIndex < 5) {
    if (digitalRead(aiuInputPins[aiuInputIndex]) == HIGH)
    {
        digitalWrite(aiuInputPins[aiuInputIndex + 9], LOW);
    }
    else
    {
        digitalWrite(aiuInputPins[aiuInputIndex + 9], HIGH);
    }
}
else if (outp_mode == 0x0A && aiuInputIndex > 3 && aiuInputIndex < 9) {
    if (digitalRead(aiuInputPins[aiuInputIndex]) == HIGH)
    {
        digitalWrite(aiuInputPins[aiuInputIndex + 5], LOW);
    }
    else
    {
        digitalWrite(aiuInputPins[aiuInputIndex + 5], HIGH);
    }
}
aiuInputIndex++;
if(aiuInputIndex >= NUM_AIU_INPUTS)
    aiuInputIndex = 0;
}
}
delayMicroseconds(500);
} // End loop

```