# FSCRIPT Product Spotlight #3

**December 11, 2024**

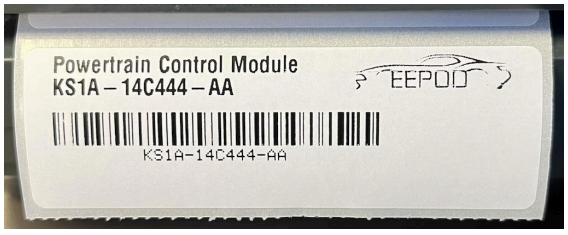
We are continually adding new features to the FSCRIPT compiler programming language to allow our customers to perform complex tasks much easier.  In this product spotlight we will introduce a few of the latest new features: Barcode label printing, FlexStation-IOT power supply control and USB command server interface.


**EEPod FlexStation-IOT with Label Printer and ECU POGO Pod Attachments (total cost ~$5K, depending on ECU pod type)**

# Barcode Label Printing

We have added support to connect a MyCANIC-IOT or Flexstation IOT directly to a ZPL (Zebra Print Language) compatible barcode label printer. Using our new RS232 adapter cable ($125.00USD), you can connect the MyCANIC-IOT or Flexstation-IOT directly to the RS232 port of the printer and then add ZPL commands to any script to print any custom label, including barcodes and logos. The ZPL commands are standard, with the exception that the data field (^FD…) can be filled with data read from a vehicle/ECU. Below is an example script for creating a label with both a logo and barcode (label commands highlighted in red).

```
VERSION "DBG20"
FW_VERSION 53.16
; Start the log file
LOG FILE "ZPL.LOG"
LOG NEW RTC
LOG ADD SERNUM
HLCD 1 1 " ZPL Label Test"
LED RED OFF
LED GREEN OFF
DELAY 500
; Initialize the HSCAN network
NET CAN_HS 11 500000
; Start the diagnostic filter for the PCM
DIAG ISO14229 0x7E0 0x7E8
LCD 3 1 ""
LCD 3 1 "Reading F188"
; Read the F188 part number from a module
REQUEST 0x22 0xF1 0x88
LCD 4 1 "Printing.."
; Start the label
ZPL "^XA"
; Position the first field at X=50, Y=20
ZPL "^FO50,20"
; Set font (A0), normal orientation (N), height=40, width=40
ZPL "^A0N,40,40"
; Print the description text
ZPL "^FDPowertrain Control Module^FS"
; Add the part number to the label at a position of 50 dots from
; left edge and 60 dots from top with Font 0 with a height of 40 dots
; Print the F188 part number from above as text
ZPL "^FO50,60^A0,40^FDTEXT[7,14]^FS"
; Print the F188 part number as a barcode
ZPL "^FO50,120^BY3^BCN,60,,,,A^FDTEXT[7,14]^FS"
; Position the logo
ZPL "^FO550,20"
; Print the EEPod logo
ZPL "^GFA,9600,9600,40,"
ZPL "gY017MFC,gX07FEA924AB7FF8,gW07EO01FFE,"
ZPL "gV03CR0IF,gU01EJ057JF4I07FF,gU0EI0IFEAA57FEI07FF,"
ZPL "gT07001FDM01FCI07FF,gS01800F8O03FJ0FFE,"
ZPL "gS0E0038Q0F8I01FFE,gR03I0ER07CJ01FFE,"
ZPL "gR0C0038R03EK03FFE,gQ02I06S01FL07FFE,"
ZPL "gQ080018T0F8K01IFE,gT02T03FJ01LFC,"
ZPL "gT04S01FEI03PF4,hN0FF8001SFD,hM0FF8I0IFAO02BFE8,"
ZPL "g016IFDB52W0FFCI07FC,X0BVFEAAL05FFAI03F8,"
ZPL "V05gOF8J0FC,U0KFA8M0224JAELFE8K03CT051,"
ZPL "T0JFgR0ET01IFD5,S07FF4gR03V0F8,R07FEgS01CV0F8,"
ZPL "Q03FEgT02N04N07C,P01FCL0AA8gS07IF8L03C,"
ZPL "P0FCL0JFgR03KFL01E,O078L03IF8gQ01FFEAFF8K01F808,"
ZPL "N038M0FE801JFC3JF83JF001JF803JFE3F8I03EL0JF8,"
ZPL "M01CM03EI01JFC1JF83JFC03JFE03KFBCK078K07IFC,"
ZPL "M04N078I01JFC3JF83JFE07KF03KF8L01CK01BDFC,"
ZPL "V0EJ01EJ01CI0878003E0F8002F03CI0BCM06N01C,"
ZPL "U018J01EJ03CJ038I0E0FJ0F038I03CM03N01E,"
ZPL "M02FCK03L0EJ01CJ038I0E0EJ07038I01CM018N0E,"
ZPL "L03IFK06K01EJ03CJ038I0F0FJ0783CI01CN08N0F,"
ZPL "O0F8J04K01EJ01CJ038I0E0EJ07038I01EN04N0F,"
ZPL "O03CJ08K01EJ03CJ078I0E0FJ07038I01CN02N07,"
ZPL "O03EI01L01EJ01CJ038I0F0EJ0783CI01CW078,"
ZPL "P0EQ0EJ03CJ038I0E0FJ07038I01CI0186Q07,"
ZPL "P0FP01EJ01CJ078I0E0EJ07038I01CW078,"
ZPL "P078O01E55683D55B038001E0FJ0783CI01EW0F,"
ZPL "P07CO01JFC3JF839IFE0EJ07038I01CV01F,"
ZPL "O09FCP0JF81JF079IFC0FJ07038I01CV01E,"
ZPL "L03JF8O01JF83JF839IF80EJ0783CI01CV07C,"
ZPL "L03IFBP01EJ01CJ038IA00FJ07038I01CV0F8,"
ZPL "L038S01EJ03CJ038K0EJ07038I01EU01F,"
ZPL "L018S01EJ01CJ038K0FJ0783CI01CU03E,"
ZPL "L01CT0EJ03CJ078K0EJ07038I01CU07C,"
ZPL "M0ES01EJ01CJ038K0FJ07038I01CU0F8,"
ZPL "M0FS01EJ03CJ038K0EJ0783CI01CT01F8,"
ZPL "M078R01EJ01CJ038K0F8001F038I07CS07E,"
ZPL "M03CS0EJ03CJ038K0EJ07038I01CT0FC,"
ZPL "M01ER01EJ01CJ038K0FJ0783CI01CS01F8,"
ZPL "M01ER01EJ03CJ078K0FJ0F038I03CS03F,"
ZPL "M03ER01EJ01CJ038K0F8001F038I07CS07E,"
ZPL "M03ER01JFC3JF838K07KF03DJFCS0FC,"
ZPL "M07CS0JFC1JF838K07JFE039JF8R03FC,"
ZPL "M0F8R01JFC3JF878K01JF803DIFES07F8,"
ZPL "L01FhX0FF8,L03ChX0FF8,M08hW01FE,iL06,"
; Finish the label and start printing
ZPL "^XZ"
LCD 8 1 "Press any key..."
WAITKEY ANY
EXIT
```




**Actual Printed Label**

# Flexstation-IOT Power Supply Control

We have implemented new FSCRIPT features that allow you to set the voltage, read the current and operate a switched output (ignition line) from a script. This allows for extensive ECU testing/validation without a PC connected. The script below is an example of performing various ECU validation tests using these new power supply features (highlighted in red).

```
FW_VERSION 53.16
VERSION "DBG20"
; Start the log file
LOG FILE "ECUTest.LOG"
; Calibrate the FlexStation-IOT power supply
LCD 2 1 "Calibrating..."
SET_VALUE AMPS 0
NEXT_TEST:
LCD 1 1 ""
HLCD 1 1 " ECUTest v1.00  "
; Connect to the appropriate network
NET CAN_HS 11 500000
; Setup the ECU diagnostic filter
DIAG ISO14229 0x707 0x70F
; Start each log record with the UTC date/time and
MyCANIC-IOT S/N
LOG NEW RTC
LOG ADD "MyCANIC-IOT S/N"
LOG ADD SERNUM
; Turn on power supply to 12.000VDC
SET_VALUE VOLTS 12000
; Turn on switched power output (ignition line)
SET_VALUE IGN_LINE 1
; Allow a moment to power up
DELAY 100
; Read the ECU serial number
LCD 3 1 "Logging info..."
REQUEST 0x22 0xF1 0x8C
IF DATA[4,1]!0x62
  LOG ADD "ERROR: No ECU detected!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: No ECU"
  LCD 4 1 "detected!"
  GOTO ERROR_EXIT:
ENDIF
; Make sure serial number is ASCII numeric value (check
first digit)
IF DATA[7,1]<0x30
  LOG ADD "ERROR: Invalid ECU S/N!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: Invalid"
  LCD 4 1 "ECU S/N!"
  GOTO ERROR_EXIT:
ENDIF
IF DATA[7,1]>0x39
  LOG ADD "ERROR: Invalid ECU S/N!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: Invalid"
  LCD 4 1 "ECU S/N!"
ENDIF
; Record ECU serial number
LOG ADD "F18C"
LOG ADD TEXT[7,16]
; Read and record ECU part number
REQUEST 0x22 0xF1 0x88
LOG ADD "F188"
LOG ADD TEXT[7,16]
; Turn off ignition line and allow time for ECU to fall asleep
LCD 4 1 "Measuring OFF mA"
SET_VALUE IGN_LINE 0
DELAY 1000
; Read and check the ECU off current
LOG ADD "OFF current(mA)"
GETTIME
DO
  VAR[0]=AMPS
  IF VAR[0]<10
    ENDDO
  ENDIF
WHILE TIME<2000
LOG ADD VAR[0]
IF VAR[0]>10
  LOG ADD "ERROR: OFF current too high!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: OFF"
  LCD 4 1 "current is too"
  LCD 5 1 "high!(       mA)"
  LCD 5 7 VAR[0]
  GOTO ERROR_EXIT:
ENDIF
; Turn on ignition line and allow time to wake up
LCD 5 1 "Measuring ON mA"
SET_VALUE IGN_LINE 1
DELAY 500
; Read and check the on current
LOG ADD "ON current(mA)"
VAR[0]=AMPS
LOG ADD VAR[0]
IF VAR[0]<70
  LOG ADD "ERROR: ON current too low!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: ON"
  LCD 4 1 "current is too"
  LCD 5 1 "low! (       mA)"
  LCD 5 7 VAR[0]
  GOTO ERROR_EXIT:
ENDIF
IF VAR[0]>130
  LOG ADD "ERROR: ON current too high!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: ON"
  LCD 4 1 "current is too"
  LCD 5 1 "high!(        A)"
  LCD 5 7 VAR[0]
  GOTO ERROR_EXIT:
ENDIF
```

```
; Read and check DTCs
LCD 6 1 "Checking DTCs..."
IF DTC!0
  LOG ADD DTC
  LOG ADD "ERROR: DTCs Set!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: DTCs Set!"
  LCD 4 1 "#DTCs:"
  LCD 4 7 DTC
  GOTO ERROR_EXIT:
ELSE
  LOG ADD "No DTCs"
ENDIF
; Verify normal mode / broadcast messages
LCD 7 1 "Checking Msgs..."
LOG ADD "MSG 0x688(msec)"
GETTIME
DO
  READ MSG 0x688
  IF DATA[2,2]=0x688
    GETTIME
    DO
      READ MSG 0x688
      IF DATA[2,2]=0x688
        VAR[1]=TIME
        ENDDO
      ENDIF
    WHILE TIME<2000
    ENDDO
  ENDIF
WHILE TIME<2000
IF DATA[2,2]!0x688
  LOG ADD "ERROR: ID 0x688 not received!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: ID 0x688"
  LCD 4 1 "not received!"
  GOTO ERROR_EXIT:
ENDIF
; Check the timing between messages (1000msec)
LOG ADD VAR[1]
IF VAR[1]<998
  LOG ADD "ERROR: ID 0x688 too fast!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: ID 0x688"
  LCD 4 1 "too fast!"
  GOTO ERROR_EXIT:
ENDIF
IF VAR[1]>1002
  LOG ADD "ERROR: ID 0x688 too slow!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: ID 0x688"
  LCD 4 1 "too slow!"
  GOTO ERROR_EXIT:
ENDIF
; Check data in the 0x688 message
LOG ADD "S/W VERSION"
LOG ADD HEX[10,2]
IF DATA[10,2]!0x119
  LOG ADD "ERROR: S/W Version!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: S/W VER"
  LCD 4 1 "not v1.25!"
  GOTO ERROR_EXIT:
ENDIF
LOG ADD "VBAT"
LOG ADD DATA[4,2]
IF DATA[4,2]<11700
  LOG ADD "ERROR: VBAT Low!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: VBAT Low!"
  LCD 4 1 DATA[4,2]
  GOTO ERROR_EXIT:
ENDIF
IF DATA[4,2]>12300
  LOG ADD "ERROR: VBAT High!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: VBAT Hi!"
  LCD 4 1 DATA[4,2]
  GOTO ERROR_EXIT:
ENDIF
LOG ADD "IGN"
LOG ADD DATA[6,2]
IF DATA[6,2]<11700
  LOG ADD "ERROR: IGN Low!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: IGN Low!"
  LCD 4 1 DATA[6,2]
  GOTO ERROR_EXIT:
ENDIF
IF DATA[6,2]>12300
  LOG ADD "ERROR: IGN High!"
  LCD 2 1 ""
  LCD 3 1 "ERROR: IGN High!"
  LCD 6 1 DATA[6,2]
  GOTO ERROR_EXIT:
ENDIF
GETTIME
; Turn off ignition line and power supply
SET_VALUE IGN_LINE 0
SET_VALUE VOLTS 0
; Test complete and successful
LED GREEN ON
LOG ADD "Factory Test PASSED!"
LCD 2 1 ""
LCD 3 1 "TEST COMPLETED"
LCD 4 1 "SUCCESSFULLY!"
LCD 6 1 "Press ENTER for"
LCD 7 1 "next test or *"
LCD 8 1 "to exit."
WAITKEY ANY
LED GREEN OFF
IF KEY=ENTER
  GOTO NEXT_TEST:
ELSE
  EXIT
ENDIF
ERROR_EXIT:
LED RED ON
LCD 8 1 "Press any key..."
WAITKEY ANY
LED RED OFF
EXIT
```

# USB Command Server Interface

The MyCANIC-IOT USB interface has been added as a protocol type to be used with the READ and WRITE commands, allowing the user to control the operation of a script from a host PC. Using this feature along with our Command Server Windows PC application, it is possible to control several MyCANIC-IOTs from a single PC in a manufacturing environment. Below is an example that uses a master script that calls a programming script based on the command from the host PC. The host PC commands are highlighted in red.

**Master Script (AUTOEXEC.FSF)**

```
FW_VERSION 53.16
VERSION "DBG20"
HLCD 1 1 "USB Cmd    v1.00"
LCD 8 1 "Press * to exit."
; Read and process commands from the USB port
DO
 ; Read from the USB port
 READ USB
 ; Check if a command was received
 IF DATA[0,1]!0x00
  ; Show the command received on the LCD
  LCD 2 1 TEXT[0,24]
  DELAY 500
  ; Check for a PROG command
  IF TEXT[0,4]="PROG"
   LCD 3 1 "Programming..."
   DELAY 500
   ; Check if programming ECU#1
   IF TEXT[5,2]="ECU#1"
    LCD 4 1 "ECU #1..."
    ; Call the script to program ECU1
    EXEC "ECU1.FSF"
    EXIT
   ENDIF
   ; Check if programming ECU#2
   IF TEXT[5,2]="ECU#2"
    LCD 4 1 "ECU #2..."
    DELAY 500
    ; Call the script to program ECU2
    EXEC "ECU2.FSF"
    EXIT
   ENDIF
  ENDIF
 ENDIF
 ; Check if any pending IOT requests
 ; This allows IOT server requests to be performed when
 ; not performing programming tasks.  In a production line,
 ;  you will want to take care to only perform IOT updates
 ;  when the production line is not running (e.g. between
 ;  shifts or at night).
 IF IOT!0
  EXIT
 ENDIF
 ; Check for a key press in case the user wants to exit
 GETKEY
WHILE KEY!ESCAPE
EXIT
```

**ECU Programming Scripts (ECU1.FSF and ECU2.FSF)**

```
FW_VERSION 53.16
VERSION "DBG20"
LCD 1 1 "ECU #1     v1.00"

; Start the log file
LOG FILE "ECU1.LOG"
LOG NEW RTC
LOG ADD SERNUM
LOG ADD "ECU1 v1.00"

; Do the programming...
DELAY 2000

; Write the last log entry as response to command server
WRITE USB
EXIT
```

```
FW_VERSION 53.16
VERSION "DBG20"
LCD 1 1 "ECU #2     v1.00"

; Start the log file
LOG FILE "ECU2.LOG"
LOG NEW RTC
LOG ADD SERNUM
LOG ADD "ECU2 v1.00"

; Do the programming...
DELAY 2000

; Write the last log entry as response to command server
WRITE USB
EXIT
```