FlexCase

User Manual

Table of Contents

Table of Contents

Datasheet

Changelog

Getting Started
- Overview
- Software Recommendations
  - MCU
  - MPU
- Getting Help
- Modifying Cinch Connectors Without Tools
  - Removing the Secondary Lock
  - Removing the Terminal
- Opening Cinch Enclosure Without Tools

Pins and Ports

Integration Recommendations

MCU Considerations
- Programming the MCU
  - Setting up the debug environment
- Modules of the MCU
  - Power
    - Power Keep On
    - Supply Voltage Read
    - Current Read
  - Analog Reads
    - Pull-Ups
  - CAN
    - Enable
    - Terminating Resistors
  - Digital Inputs
  - Digital Outputs
    - Introduction
    - Configuration and Use
    - Pulse Width Modulation (PWM)
  - Accelerometer

- Buzzer
- MPU Considerations
  - Power Management
    - Audesse Included Support
    - Only Boot When Required
    - Graceful Shutdown
    - Harden the MPU
  - Flashing the MCU
    - Audesse Included Support
  - Reflashing the OS
  - USB A Slot
  - Potentially Useful Software
- MCU-MPU Collaboration
  - MPU Reset
    - Suggested Uses
  - SPI
    - Suggested Uses
    - Setup
- IoT Considerations
  - Supported Hardware
    - Audesse Included Support
  - Alternate Connectivity Options
- Future Works
- Compatible Parts
- Known Issues
  - Issue 1 - High Side PWM Keep-off

Datasheet

Processor & High Reliability Interfaces	
Processor	**NXP FS32K144HAT0MLLT** General purpose automotive and high reliability (ASIL B Capable) Programmability: Via JTAG, Via (W)LAN Software: MATLAB Simulink, NXP S32DS platform (C with SDK) Debugging and Logging: Freemaster, Onboard Storage
Power	8 - 36V Ignition pin (active high) Internal voltage and current sense Undervoltage, Overvoltage, Reverse Polarity protection Consumption: 2.5W nominal. 0.025W standby.
Analog Inputs	6 x 0 - 36V
Digital Inputs	6 active low (default)
Digital OR PWM Outputs	6 x 3A High or Low side drive 0 - 100 kHz
Digital Outputs	10 x 3A High or Low side drive
Indicator and Sensors	1 red LED - Power Good Ethernet LEDs Accelerometer Buzzer (user controllable)
CAN	1 CAN (FD) 1 CAN J1939 Supported Terminating resistors (default present, removable)

Add Ons	
Linux Co-Processor	**Raspberry Pi CM3+ 8/16/32 GB via SODIMM** Processor reprogramming via OpenSDA SPI connection to processor Ethernet SD card slot (expandable storage) USB A slot (Add camera, flash storage, Wifi dongle, etc) Note: Supports OS flashing onboard
Remote Connectivity	**Nimbelink Skywire Modem via headers** LTE-M/NB-IoT + GPS or CAT 1/2/3/4 **Audesse Custom Interfaces via headers** Wi-Fi Generic USB for ZigBee, Bluetooth, LoRa, Sigfox
Specifications	
Physical	13 x 12 x 4.3 cm & 0.3 kg IP67 -40°C to 80°C (Processor rated to 125°C)
Connectors	Cinch 5810130029 and 5810118023
Field Readiness	Deployable with production grade code using supported development environments
EMC Testing**	Radiated and conducted emissions. Immunity to conducted transients on power leads Immunity to electrostatic discharge Immunity to electromagnetic fields Immunity to radiated electromagnetic fields

** EMC has been completed for previous automotive applications but is specific to YOUR application and will have to be reconducted before legal sale. Reach out for more details.

Changelog

Name	Date	Comment
Baseline	July 25, 2020	Begin changelog
First friendly version	September 1, 2020	Rev1.0 complete

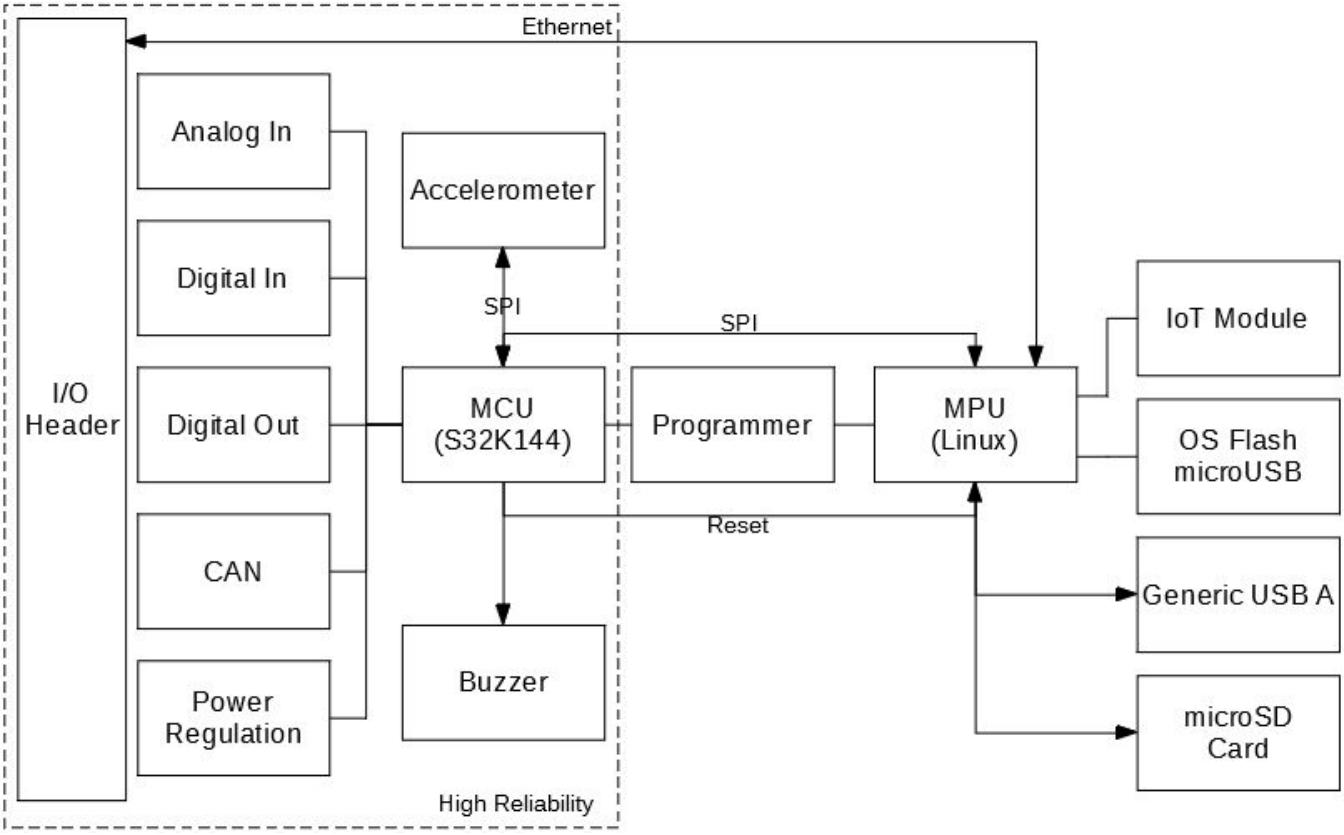
Getting Started

For first-time users, please follow the Quick-Start Guide to hook up the FlexCase and flash a program.

Overview

This product is designed to be an easy to prototype and integrate Electronic Control Unit (ECU) for automotive general purpose and high reliability applications. When required, the product can be equipped with a full linux co-processor for edge computing applications and many connectivity options (Wifi, Bluetooth, Cellular, GPS).

The functional breakdown of the control unit is shown below



The dashed lines show where safety critical applications can sit. You can use the non-safety critical components in the control system but they should NOT be relied upon. You can use the MCU to safety check any inputs from non-safety critical components.

Software Recommendations

The following free or low cost softwares are recommended for developing on the ECU.

MCU

The MCU software support is summarized and updated on the S32K1xx product page. The current methods are summarized below.

S32DS-ARM: S32 Design Studio for Arm (link) is the fundamental development for the MCU coding environment. It is complementary and gives you the most control over the actions of the MCU. Some insights for using the IDE

- The SDK functions should be used with minimal logic being written for your application
- ProcessorExpert is a great tool for configuring the hardware of the device
- This is the tool you want to use to really understand what is going on

FreeMASTER (link) is a great included tool used to live monitor and visualize states and variables in the MCU.

Model-Based Design Toolbox (link) is another complimentary software tool that lets you code the MCU in MATLAB Simulink. Some insights for using this toolbox

- It effectively implements most the SDK functions found in S32DS as blocks
- It is great for quick prototyping and can be used to generate near production ready code
- For certain reliability, the generated code should be imported into S32DS for manual analysis

Note: In order to use the toolbox you will **need** to have the following Mathworks Products

- MATLAB
- Simulink
- MATLAB Coder
- Simulink Coder
- Embedded Coder

MPU

The best resource for working with the MPU is the Raspberry Pi community. Almost any question you have has already been answered. Our shortlist of programs that work well with the ECU are below.

OS - We recommend the Raspberry Pi OS Lite as it has the most support. Considerable alternatives include Ubuntu Core and BalenaOS.

OS Flashing - For quickly putting a new OS on the MPU.
We recommend balenaEtcher. It's an extremely simple tool for flashing most images.

SSH - For communicating with the MPU over the network.

For Windows systems we recommend PuTTY for terminal communication, WinSCP for file transfers, and Advanced IP Scanner for finding the internal IP address of the ECU on the network.

SSH Over The Internet - If your ECU will be connected to the cellular modem or a network that is connected to the internet you can take advantage of the service Dataplicity. It will allow you to SSH into the MPU from anywhere using the internet.

USBoverIP - For connecting USB devices on the ECU to the host PC while debugging (MPU programmer chip, IoT modem, USB A plug)
We recommend Virtualhere installing a Linux server on the MPU (Linux ARM) and the client software on the development machine.

Getting Help

There are thriving communities to help you debug your work. Depending on the area we recommend the following resources. The ECU is similar to the evaluation boards so most advice written for NXPs boards is applicable to the ECU (with different pins and ports connected).

Similar equivalents

- The MCU side of the ECU is similar to the S32K144EVB so most guides are applicable to the ECU (with different pins and ports). *Please see our software support for preconfigured templates to help you get coding faster.*
- The MPU is effectively a Raspberry Pi 3B+ in a compact form, so most guides for the Raspberry Pi will work. *Purchasing the onboard computer includes logging and power management functionality built in. Please contact us for more details.*
- The IoT side is effectively a built in Nimbelink Raspberry Pi Skywire Adapter. This means that any of the Nimbelink pre certified modems will work directly with the ECU. *Purchasing the IoT package includes simplified drivers including a global SIM.*

MCU coded in S32DS - The S32 Design Studio community. Posted questions are usually answered within 24 hours.

MCU coded in MATLAB - The NXP Model-Based Design Toolbox community.

MATLAB functions (not NXP related) - The MATLAB documentation is very informative.

MPU Coding - Googling “[Problem] raspberry pi” is often the fastest way.

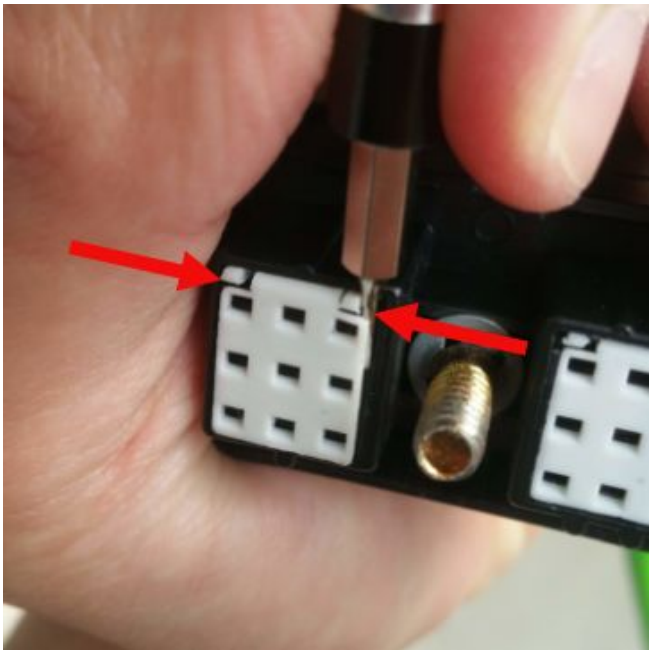
IoT Functions - Nimbelink has fast support for implementing their modules. Guides for each of their products exist on their respective pages

Modifying Cinch Connectors Without Tools

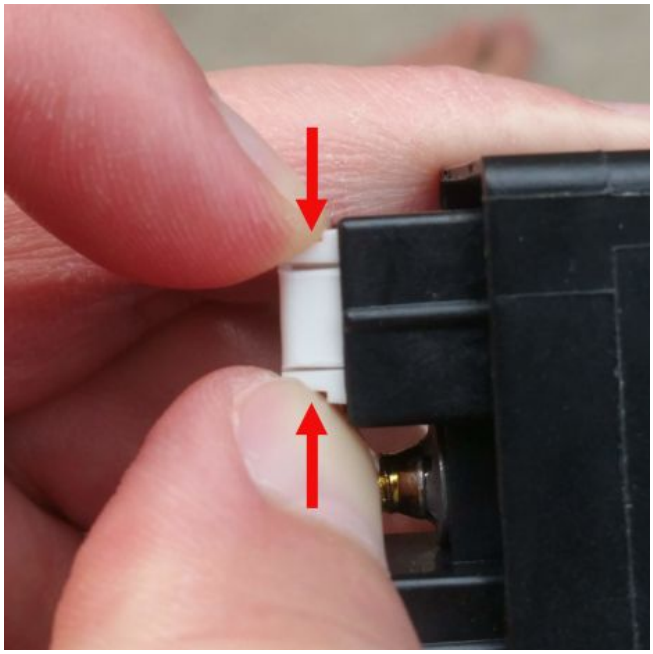
Buying the OEM tools for modifying the connectors is recommended (MPN 599 11 11 628 and 599 01 18 920), but the harness can be re-pinned with a small, 1mm wide, flathead screwdriver if needed.

Removing the Secondary Lock

The secondary lock can be removed by following the two stage process shown below:



Stage 1: The secondary lock can be removed by pinching the tabs inward as shown by the arrows. They can be done one at a time if after pinching inward, the lock is pulled out of the connector slightly so it doesn't relatch.

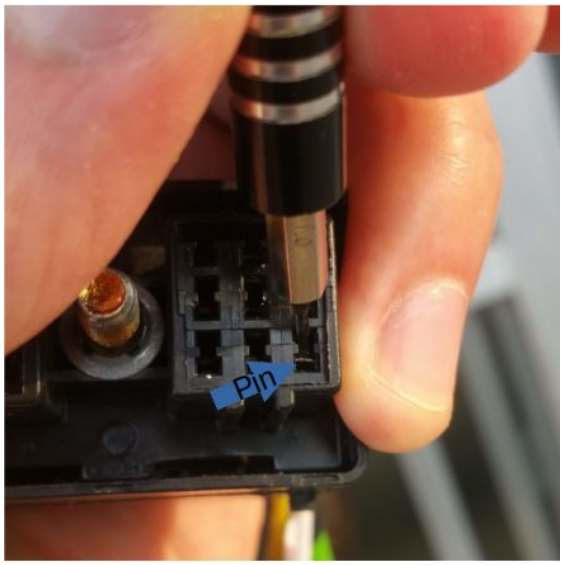


Stage 2: You can then pinch the connector as shown with your fingers to fully remove the secondary lock.

Removing the Terminal

There is a tab on the top of each pin (shown in photo) that holds it in place in the connector. By using a flathead screwdriver as shown the tab can be moved out of the way and the pin removed

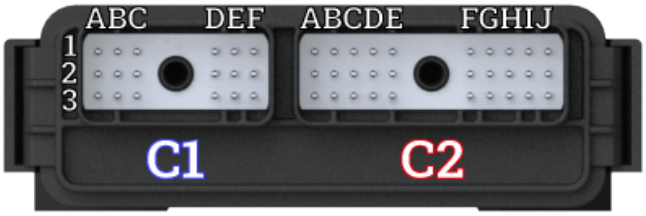
1. Slide the flat head along the top of the pin until you encounter some resistance. This is the plastic tab.
2. Gently slide the flathead between the pin and the tab
3. Lift the tab up by pivoting the flat head against the pin.
4. Gently pull the wire, the pin should slide out of the connector



Opening Cinch Enclosure Without Tools

The OEM tool is recommended (MPN 5991111440), but the enclosure can be opened using multiple flathead screwdrivers in a pinch.

Pins and Ports



Color legend: 18 Pin header, 30 pin header, internal pin

Pin #	Function	Port	Application Comment
Power			
C2-H1	+V_BATT	N/A	8-36V
N/A	Batt V Meas	PTA6/ADC0_SE2	Measures input voltage
C2-H3	IGNITION EXT	PTE13	Detect V+ (Digital Input)
N/A	IGNITION KEEPON	PTD7	Parallel to IGN (Digital Output)
C2-H2	GND_IN	N/A	Grounds are common
C2-G1	GND_IN	N/A	Grounds are common
C2-D1	+5V0_EXT	N/A	2A Max (fused to 8A)
N/A	Current Monitor	PTD4/ADC1_SE6	No SDK
CAN			
C1-A1	CAN0 H	PTB1	FD capable
C1-B1	CAN0 L	PTB0	FD capable
N/A	CAN0 Enable	PTC10	Pull low to enable
C1-C2	CAN1 H	PTA13	No Comment
C1-C3	CAN1 L	PTA12	No Comment
N/A	CAN1 Enable	PTA14	Pull low to enable
Ethernet			
C1-A2	ETH RX N		568B Orange
C1-A3	ETH RX p		568B Orange White
C1-B2	ETH TX N		568B Green
C1-B3	ETH TX P		568B Green White
C2-E2	ETH RX+ SPARE	Not Connected	Reserved for gigabit
C2-E3	ETH RX- SPARE	Not Connected	Reserved for gigabit

C2-G2	ETH TX- SPARE	Not Connected	Reserved for gigabit
C2-G3	ETH TX+ SPARE	Not Connected	Reserved for gigabit
Digital Outputs			
	Digital Output Bank 1		
C2-A3	+VCC_DO_1	N/A	Supplies DO 1 - 5
C1-C1	DO1_EXT	PTD15/FTM0_CH0 PTD16/FTM0_CH1	High Side (GPIO) Low Side (GPIO/PWM)
C1-D3	DO2_EXT	PTD8/FTM1_CH4 PTD9/FTM1_CH5	High Side (GPIO) Low Side (GPIO/PWM)
C1-D2	DO3_EXT	PTE15/FTM2_CH6 PTE16/FTM2_CH7	High Side (GPIO) Low Side (GPIO/PWM)
C1-D1	DO4_EXT	PTC2/FTM0_CH2 PTC3/FTM0_CH3	High Side (GPIO) Low Side (GPIO/PWM)
C1-E3	DO5_EXT	PTB2/FTM1_CH0 PTB3/FTM1_CH1	High Side (GPIO) Low Side (GPIO/PWM)
	Digital Output Bank 3		
C2-E1	+VCC_DO_3	N/A	Supplies DO 6 - 10
C1-E2	DO6_EXT	PTD0/FTM2_CH0 PTD1/FTM2_CH1	High Side (GPIO) Low Side (GPIO/PWM)
C1-E1	DO7 EXT	PTC12/FTM3_CH6 PTC13/FTM3_CH7	High Side (GPIO) Low Side (GPIO/PWM)
C1-F3	DO8 EXT	PTD2/FTM3_CH4 PTD3/FTM3_CH5	High Side (GPIO) Low Side (GPIO/PWM)
C1-F2	DO9 EXT	PTE10/FTM2_CH4 PTE11/FTM2_CH5	High Side (GPIO) Low Side (GPIO/PWM)
C1-F1	DO10 EXT	PTE8/FTM0_CH6 PTE9/FTM0_CH7	High Side (GPIO) Low Side (GPIO/PWM)
	Digital Output Bank 2		
C2-A1	+VCC_DO_2	N/A	Supplies DO 11 - 16
C2-A2	DO11 EXT	PTC14/FTM1_CH2 PTC15/FTM1_CH3	High Side (GPIO/PWM) Low Side (GPIO/PWM)
C2-B3	DO12 EXT	PTB10/FTM3_CH2 PTB11/FTM3_CH3	High Side (GPIO/PWM) Low Side (GPIO/PWM)
C2-B2	DO13 EXT	PTE4/FTM2_CH2 PTE5/FTM2_CH3	High Side (GPIO/PWM) Low Side (GPIO/PWM)
C2-B1	DO14 EXT	PTB4/FTM0_CH4 PTB5/FTM0_CH5	High Side (GPIO/PWM) Low Side (GPIO/PWM)
C2-C3	DO15 EXT	PTC0/FTM1_CH6 PTC1/FTM1_CH7	High Side (GPIO/PWM) Low Side (GPIO/PWM)

C2-C1	DO16 EXT	PTB8/FTM3_CH0 PTB9/FTM3_CH1	High Side (GPIO/PWM) Low Side (GPIO/PWM)
Digital Inputs / Quadrature Inputs			
C2-C2	QUAD2 B EXT	PTD10/FTM2_QD_PHB	Digital Input
C2-D2	QUAD2 I EXT	PTD12	Digital Input
C2-D3	QUAD2 A EXT	PTD11/FTM2_QD_PHA	Digital Input
C2-F1	QUAD1 B EXT	PTC6/FTM1_QD_PHB	Digital Input
C2-F2	QUAD1 I EXT	PTA0/FXIO_D2	Digital Input
C2-F3	QUAD1 A EXT	PTC7/FTM1_QD_PHA	Digital Input
Analog Inputs			
C2-J1	ADC 6 EXT	PTB12/ADC1_SE7	0-36V or (NC vs GND)
N/A	ADC 6 PULL EN	PTE7	Pull up to V+
C2-J2	ADC 5 EXT	PTA15/ADC1_SE12	0-36V or (NC vs GND)
N/A	ADC 5 PULL EN	PTD17	Pull up to V+
C2-J3	ADC 4 EXT	PTA16/ADC1_SE13	0-36V or (NC vs GND)
N/A	ADC 4 PULL EN	PTC8	Pull up to V+
C2-K1	ADC 3 EXT	PTE6/ADC1_SE11	0-36V or (NC vs GND)
N/A	ADC 3 PULL EN	PTE3	Pull up to V+
C2-K2	ADC 2 EXT	PTA1/ADC0_SE1	0-36V or (NC vs GND)
N/A	ADC 2 PULL EN	PTE12	Pull up to V+
C2-K3	ADC 1 EXT	PTB13/ADC1_SE8	0-36V or (NC vs GND)
N/A	ADC 1 PULL EN	PTC9	Pull up to V+
Internals			
N/A	Buzzer	PTD5/FTM2_CH3	3-8 kHz Recommended
	MPU Reset	PTA17	Momentary active high GPO
	MCU-MPU SPI	SPI1	See MCU-MPU Collaboration
	Accelerometer	SPI0	Modules of MCU - Accelerometer
IoT Control (Controlled by MPU)			
	Skywire Pin 5	Output - GPIO4	Active high GND reset
	Skywire Pin 6	Output - GPIO1	Active high supply 3.3V
	Skywire Pin 20	Output - GPIO0	Active high GND on/off SW
	Skywire Pin 13	Input - GPIO7	Low=Modem On;High=Modem off
	Skywire Pin 14	Output - GPIO2	High=Enable;Low=PwrSavings
	Skywire Pin 2 & 3	TX=GPIO14; RX=GPIO15	
	Skywire Pin 7& 8	USB	
	Skywire Pin 9	GPIO6	

	Skywire Pin 12	GPIO16	
	Skywire Pin 16	GPIO17	

Integration Recommendations

These are the recommendations we make for integrating the ECU with the most success

- Connect +V_BATT and GND to **always on, unswitched** power and use the switched signal for the Ignition signal. This has multiple benefits.
    - The switched signal can be rated for very low current, typically about 1mA of current
    - The ECU can be gracefully powered down with software. The ECU can be setup to not immediately powerdown when the ignition signal is removed. Rather it can take its time to end processes and more smoothly shutdown
- Prefer LAN instead of WLAN, especially for debugging.
    - LAN provides a much more reliable experience, whereas WLAN can be choppy
    - The M12 port can be panel mounted and waterproofed if required

MCU Considerations

Programming the MCU

Setting up the debug environment

We recommend using CAN for debugging if possible, packing up status signals into a single message and broadcasting it every loop for tracking. Alternatively Freemaster can be used if desired

Modules of the MCU

There are many components to customize in the MCU. Read over the recommended approaches below:

Power

The MCU has control and can read the status of the main power supply for the board, the 5V DCDC.

Power Keep On

The purpose of the IGNITION KEEPON is to keep power supplied to the board after the ignition signal is removed to enable a proper shutdown sequence. Use of the IGNITION KEEPON function is as follows:

1. On boot, assert the IGNITION KEEPON (digital output) high.
2. During the algorithm continue to read IGNITION EXT (digital input). If the read returns high the ignition signal is still applied.
3. When the IGNITION EXT signal is removed (read low) start the shutdown sequence. This can involved
    a. Doing any housekeeping before shutdown (logging/sending messages)
    b. Telling the MPU to do its own shutdown routine and giving it time to do it.
    c. Making sure the control system is safe to shutdown
4. When ready the powerdown sequence can be completed by asserting the IGNITION KEEPON low.

The board is now completely powered down. It is important to note that the IGNITION KEEPON signal can only *keep on* the power system, not start it whenever desired. Therefore after a full shutdown an ignition signal is required to restart the ECU

Supply Voltage Read

The supplied voltage powering the board can be measured using Batt V Meas. This is an analog input could be used to change behaviour based on the estimated state-of-charge of the low voltage system

Current Read

The current consumption for the main DCDC can be read with Current Monitor functionality. This is an analog input that has been verified, but no SDK is currently supplied by Audesse. If required, request or profile the functionality for yourself.

Analog Reads

Most analog reads have a voltage divider that scales down the input voltage to a safe voltage for reading. The factor is 1/12th, so a 36V input will reach the MCU at 3.0V
The included signals that have this voltage divider include

- All Analog Inputs
- Batt V Meas

The equation to calculate theoretical ticks is based on

$$\frac{Resolution\,of\ ADC}{MCU\,System\,Voltage}=\frac{ADC\,Reading}{MCU\,Voltage\,Input},\ MCU\,Voltage\,Input=\frac{Input\,Voltage}{Voltage\,Divider}$$

Solving

$$ADC\,Reading=\frac{Resolution\,of\ ADC}{3300\,mV}*\frac{Input\,Voltage\,(mV)}{12}$$

The theoretical scaling factor for a 12V input calculation can be completed as

Input Voltage	Voltage Divider (1/12th input)	Ratiometric value (3.3V Vref)	Chosen Resolution	Read ticks
12000 mV	1000 mV	1000 / 3300	8 bit, -> 255	77
			10 bit -> 1023	310
			12 bit -> 4095	1240

However, there are voltage drops along the sensing line and tolerances with the components, so it is always best to calibrate the analog read for your application with a multimeter.

Pull-Ups

The analog reads have a built in pull up when required. This would allow the input circuit to be able to detect the difference between a grounded input and a not connected. At this time this feature is not fully supported by Audesse. If required for your application please make a request and we will get back to you.

CAN

CAN0 is flexible data rate capable. The CAN transceivers meet the ISO 11898-2:2016 and ISO 11898-5:2007 physical layer standards.

There are a lot of ways to run a CAN bus, the best option depends on each particular CAN application. Things to consider include

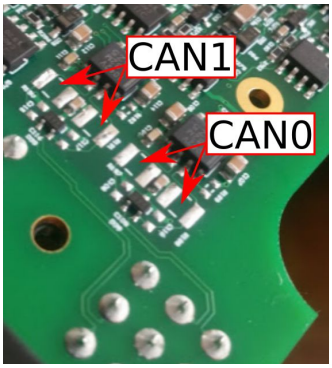
- To use FIFO (first in first out) or not
- To use blocking or non blocking for sending and receiving
- To use interrupt service routines for received CAN messages or not
- Proper use of filtering and masking messages

Enable

The CAN transceivers must be enabled in order to be used on the ECU. This can be simply achieved by driving the particular CAN enable pin, CAN# Enable, low.

Terminating Resistors

The ECU has the option for on-board terminating resistors. You can verify that your CAN resistors have been removed if your board matches the photo below, or by measuring the impedance of the header CAN pins.



Digital Inputs

The digital inputs can be used for quadrature encoders or simply as regular digital inputs.

By default, each input is pulled up to the supply voltage, but can be modified to a 5V pullup or pull down when required.

For the default configuration, the input will read high unless it is grounded, in which case it will read low.

When reconfigured with a pull down

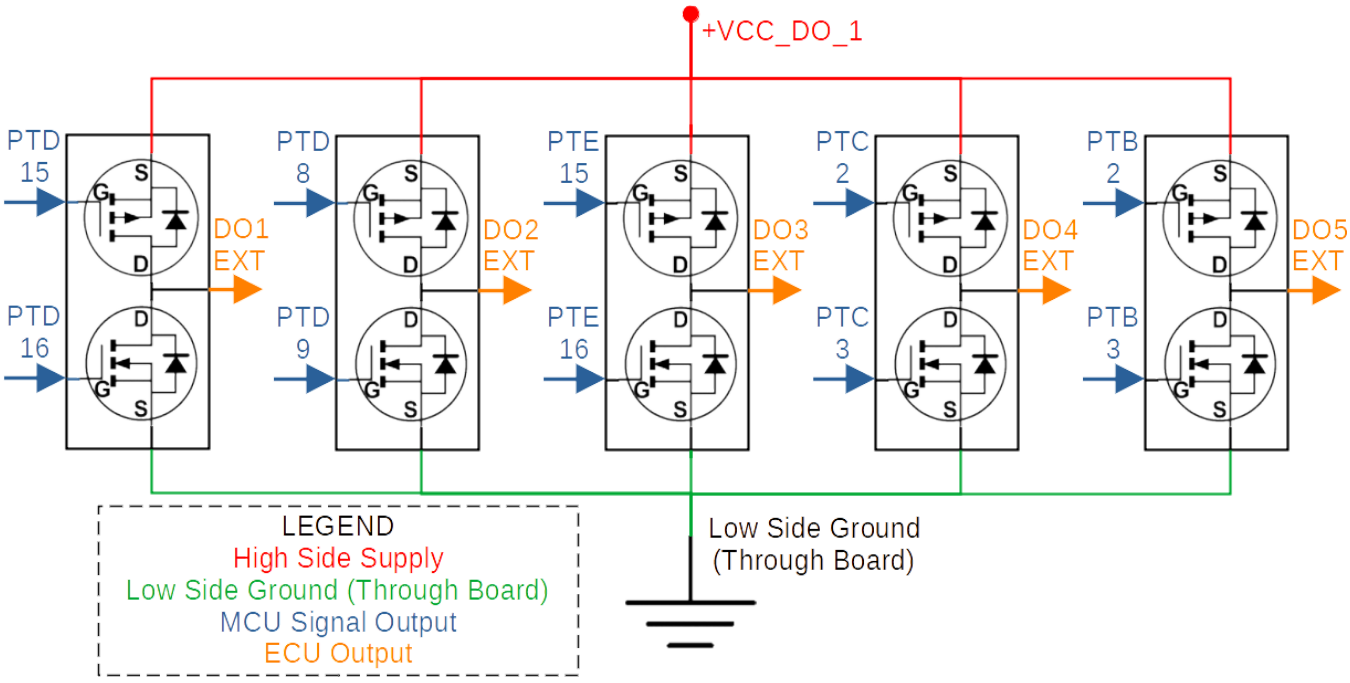
- A supplied voltage greater than 2V will result in a high reading.
- You can apply a voltage up to the maximum voltage on the ECU (36V)

Digital Outputs

Introduction

BEFORE READING THIS SECTION SEE THE PWM ISSUE IN THE KNOWN ISSUES SECTION. The digital outputs are highly versatile, but **must** be configured correctly. If configured incorrectly damage to the board may occur. Typically this will only damage a single output.

The diagram below shows the configuration for Digital Output Bank 1. Banks 2 and 3 are nearly identical in implementation.



Effectively, each output is configured as an H-bridge circuit, with the MCU having the ability to control the high and low side independently.

Our recommendation for integration is as follow:

- Whenever possible, low side drive is recommended.
- If general high side drive is required, use outputs 1 through 10
- Only use outputs 11 through 16 for high side if high side PWM is required. These outputs have the known high side issue and can be damaged if not integrated with care.

Configuration and Use

- Low side drive is **active high**.
- High side drive is **active low**. You must supply a voltage on the high side supply for high side drive to function
- **IF HIGH AND LOW SIDE DRIVE ARE ACTIVE AT THE SAME TIME THE OUTPUT WILL BE DESTROYED.**

Permissible output pin states are shown in the table below, using Digital Output 1 (DO1 EXT) as an example.

		Low Side Drive MOSFET MCU Control Pin		
		PTD16 Unconfigured /floating	**PTD16 Low**	**PTD16 High**
High Side Drive MOSFET MCU Control Pin	**PTD15 Unconfigured /floating**	Open Circuit. No connection	Open Circuit. No connection	Output connected to GND
	PTD15 Low	Output connected to +VCC	Output connected to +VCC	DANGER! CAN DAMAGE BOARD
	PTD15 High	Open Circuit. No connection	Open Circuit. No connection	Output connected to GND

Any voltage (within specified limits) can be supplied on the high side supply.

- The 5V output could be fed into the high side supply to make a 5V digital output
- A battery supply could be fed into the supply

Pulse Width Modulation (PWM)

Availability:
All low side drive output are capable of PWM
High side outputs 11 through 16 are capable of PWM

They have been tested to have good waveforms above 100 kHz. If using the PWM functionality the MCU must be configured using the Flexible Timer Module (FTM), not as GPIO using the port and pin.

Accelerometer

The manufacturer part number of the used component is AIS2DW12TR. **No SDK is supplied for its use YET**.

Its is connected using the following setup:

Function	**MCU**	**Accelerometer**
SCK	PTE0/LPSPI0_SCK	Pin 1
SS	PTA11	Pin 2
MISO	PTE1/LPSPI0_SIN	Pin 3
MOSI	PTE2/LPSPI0_SOUT	Pin 4

INT2	PTA8	Pin 11
INT1	PTA9	Pin 12

It is important to note that the slave select isn't included in the SPI0 setup, and therefore must be done manually as follows:

1. Write slave select active
2. Do SPI master transfer
3. Write slave select non-active

The following considerations were used doing the hardware bringup in S32DS

- MCU SPI settings
    - Baud rate - 5 mHz
    - PCS - PCS0
    - PCS Polarity - Active low
    - Bits/frame - 8
    - Phase - Second edge
    - ClockPolarity - Clock Active low
    - Direction - Msb first
- MCU communicates with the accelerometer via SPI in two byte messages.
    - First byte selects R/W and the register on the accelerometer
    - Second byte sends commands or receives data.
- The SPI component in processor expert has 4 options for chip select, but on the ECU they are all being used for other purposes. **Make sure you set the chip select in processor expert to a pin that is not being used when you communicate**. You must manually bit bang PTA11 to get it working (hold high, set low before communicating, set high when finished)
- Baud rate can be set within the accelerometer datasheet allowances
- Confirm communication by reading WHO_AM_I register (always readable)
- Turn on regular data communication with CTRL1 register

Buzzer

The onboard buzzer has MPN PKMCS0909E4000-R1. Its use is integrated into the empty project template.

The volume can be adjusted if required.

MPU Considerations

Power Management

Good power management is important to reduce the likelihood of filesystem corruption on the MPU. To ensure no damage always properly shutdown the MPU, such as with the command *sudo shutdown -h now.* A few methods to protect the MPU include

Audesse Included Support

When you purchase the onboard computer package we included a power management subprocess that enables auto graceful shutdown when the ignition signal is removed.

Only Boot When Required

The MCU can be commanded to hold the reset pin for the MPU until the MPU is required.

Graceful Shutdown

Use the MCU to control the power down sequence. The MCU has the capability to hold the power rail on even when the ignition signal is removed. The typical boot to shutdown process for this approach would take the form:

1. The ignition signal powers up the board
2. The MCU asserts the IGNITION KEEPON pin high. Now the board will not loose power if the ignition signal is removed.
3. The MCU watches for a removed ignition signal by reading IGNITION EXT. If removed start the shutdown sequence.
4. The MCU does any custom application shutdown tasks.
5. The MCU sends a shutdown message to the MPU and gives the MPU about 15 seconds to shutdown. This could be verified by no longer receiving any SPI communication and a drop in current consumption.
6. The MCU can then release the IGNITION KEEPON pin, removing all power from the board

The ECU can then be safely shutdown.

Harden the MPU

If a graceful shutdown isn't possible the MPU can be hardened by making the memory read only. This hasn't been tested yet, but a guide on the principle is available here.

Flashing the MCU

The MPU can be used to program the MCU over IP using OpenSDA. See the Quick-Start Guide for information on how to set that up.

Audesse Included Support

We supply the MCU and MPU ready to use this feature for those who purchase the Onboard Computer package. **If you do not purchase the package you will not be able to benefit from flashing over IP.**

Reflashing the OS

You can reflash the OS on the MPU if necessary by following the outlined steps

1. Starting with no cables plugged in and the compute module in its slot.
    a. Connect your computer USB to the microUSB on the ECU
    b. Apply power and the ignition signal to the board (can be simultaneous)
2. Run rpiBoot as defined by the Raspberry Pi foundation here.
3. Flash your preferred OS using your preferred flasher.
4. Save a blank file named “ssh” to the boot folder (no extension). SSH is disabled by default and will allow you to SSH into the device once to enable it.
5. Safe eject the compute module and unplug the microUSB cable
6. Power down the ECU and make sure you can connect to it over internet protocol. We recommend plugging the ethernet connection into a switch/router.
7. Reapply power.
8. SSH into the compute module and enable SSH through raspi-config (blank file deletes itself every power cycle so you have to enable SSH permanently)

Now you can install the software you wish on the compute module.

USB A Slot

The USB A slot on the board is connected as a slave to the MPU. Uses include debugging and added functionality:

- USB flash drives
- Wifi dongle
- Bluetooth dongle
- Camera for vision system
- Control pad for user inputs into the control system

Potentially Useful Software

We have found the following software useful in our development and you might too.

- Rclone - Used to upload log files to an online drive. Highly configurable
- MATLAB Support Package for Raspberry Pi Hardware

MCU-MPU Collaboration

MPU Reset

The MCU has the ability to reset and/or hold the MPU powered off at any time.

Suggested Uses

- You can initialize the reset pin active during boot to keep the MPU off until it should boot at a specified time later. This could be used to ensure the system is stable or conserve power until the full MPU is needed
- You can reset the MPU if the MCU determines that the processes are hanging

SPI

The MCU can communicate with the MPU using SPI for high speed communication.

Suggested Uses

You can use this communication channel for many purposes. Some ideas include:

- The MPU can send requests or commands to the MCU. The MCU can be programmed to directly follow the MPUs commands or error check them for safety reasons
- The actions and readings of the MCU can be logged by the MPU

Setup

The getting started setup for the SPI is shown in the table below.

Parameter	MCU Interface	MPU Interface
Configuration	LPSPI1 (Instance 1)	SPI2
Role	Slave	Master
MOSI	PTB15/LPSPI1_SIN	GPIO41/SPI2 MOSI (ALT4)
MISO	PTB16/LPSPI1_SOUT	GPIO40/SPI2 MISO (ALT4)
CS	PTB17/LPSPI1_PCS3	GPIO43/ SPI2 CE0 N (ALT4)
SCK	PTB14/LPSPI1_SCK	GPIO42/SPI2 SCLK (ALT4)
Software Initialization	1. Include LPSPI Config block 2. Include LPSPI Slave Transfer block	**Enable SPI** 1. Open config file: sudo nano /boot/config.txt 2. Enable SPI: Uncomment dtparam=spi=on (remove the #) 3. Enable SPI2 with 1 chip select by adding the following line : dtoverlay=spi2-1cs 4. Install spidev: link
Software setup	**Using LPSPI Configuration Block** Instance - 1 Role - [As above] Pins - [As above] CHPA - 1 CPOL - 0 PCS Polarity - Active Low Bits/frame - 8 Phase - First edge PCS - PCS3 Clock Polarity - Clock Active High Direction/Bit Order - Msb first **Using LPSPI Configuration Block** Instance - 1 PCS - 3 PCS Polarity - Active Low Transfer Mode - Non blocking	**Using Spidev** Imports - spidev Bus - 2 CS - 0 Speed - 3000000 SPI Mode - 2

IoT Considerations

Supported Hardware

While any Nimbelink modem will work on our system. We have developed easy to use drivers for the following:

Region	Modem
North America	4G LTE CAT 1 - Sierra Wireless HL7648
Europe	4G LTE CAT 4 - Telit LE910 V2

We recommend pairing the modem with the Hologram IoT Global Sim. It works almost everywhere and there are pricing tiers depending on your stage:

- Free for small data/setup
- Pay as you go for small projects
- Scaling support

Audesse Included Support

We supply easy to use drivers for those who purchase or upgrade to the IoT package. Features include

- Auto-connecting mode - All internet connectivity requirements are automatically handled
- Python modem control - You can easily control the state of the modem if your application requires it.

Alternate Connectivity Options

Endless connectivity options can be used in place of the LTE connection. Options include Wifi, Wifi Access Point, Bluetooth, LoRa, Sigfox, Zigbee, GPS, and NB-IoT.

If your application requires additional connectivity please contact us for more detail on how to achieve your project.

Future Works

If you need a feature developed reach out and well work on it.

Future work to include

- Current monitor - No SDK is developed yet but has been verified working
- Accelerometer - No SDK is developed yet but has been verified working
- Buzzer - No SDK is developed yet but has been verified working

Compatible Parts

Components recommended by Audesse that are known to work well with the FlexCase.

- M12 to RJ45 adapter - Amphenol RJS-12D04FM-LS8001

Known Issues

Issue 1 - High Side PWM Keep-off

Context: Digital outputs 11 through 16 are capable of high side PWM. This comes at a drawback, even when these pins are used for general purpose high side drive.
Probem: If a positive voltage is applied to its supply rail (+VCC_DO_2) during bootup the internal control circuitry is unable to keep up with the transient fast enough and the output “blips” high for approximately 10ms before stabilizing.
Workaround: Assuming you need PWM high side you can manually switch the supply rail after booting. The supply voltage can be applied to +VCC_DO_2 using an automotive relay using a digital output.