EAXVA03

- Data transmission interfaces:
  - 15 Ethernet Interfaces
  - 8 FPD-Link III Video Input
  - 3 USB hosts, 1 USB OTG
  - 2 CAN, 6 CAN FD
  - 4 RS-232, 1 RS-485
  - 2 LIN, 1 FlexRay
- Operating voltage: DC 9-36V
- Operation memory: 16GB
- Main chip: NVIDIA Xavier, Infineon TC297
- Storage memory: 32GB
- Calculation capability: 32TOPS (INT8)
- Dimensions: 466×250×50mm
- Operating Temperature: -25 to 85 ℃
- Humidity: 0 - 95%, no condensation
- Storage temperature: -40 to 125 ℃
- Weight: less than 5,000g

Revision History

Date	Version	Detail	Reviser
Sep. 6, 2019	V1.0	First version	David Wang
Feb. 6, 2020	V1.1	Renew product image, pinout, address, parameter description, and EcoFlash activation method	David Wang
Feb. 11, 2020	V1.2	EcoCoder-AV updated	David Wang
Feb. 25, 2020	V1.3	Add quick start chapter	David Wang
May. 11, 2020	V1.4	Address Update	Zack Li

Contact us

Web: www.ecotrons.com

Email: info@ecotrons.com

ev-support@ecotrons.com

Address: 13115 Barton Road, Ste H,

Whittier, CA, 90605, USA

Telephone: +1 562-758-3039

+1 562-713-1105

Fax: +1 562-352-0552

Contents

Chapter 1 Summary

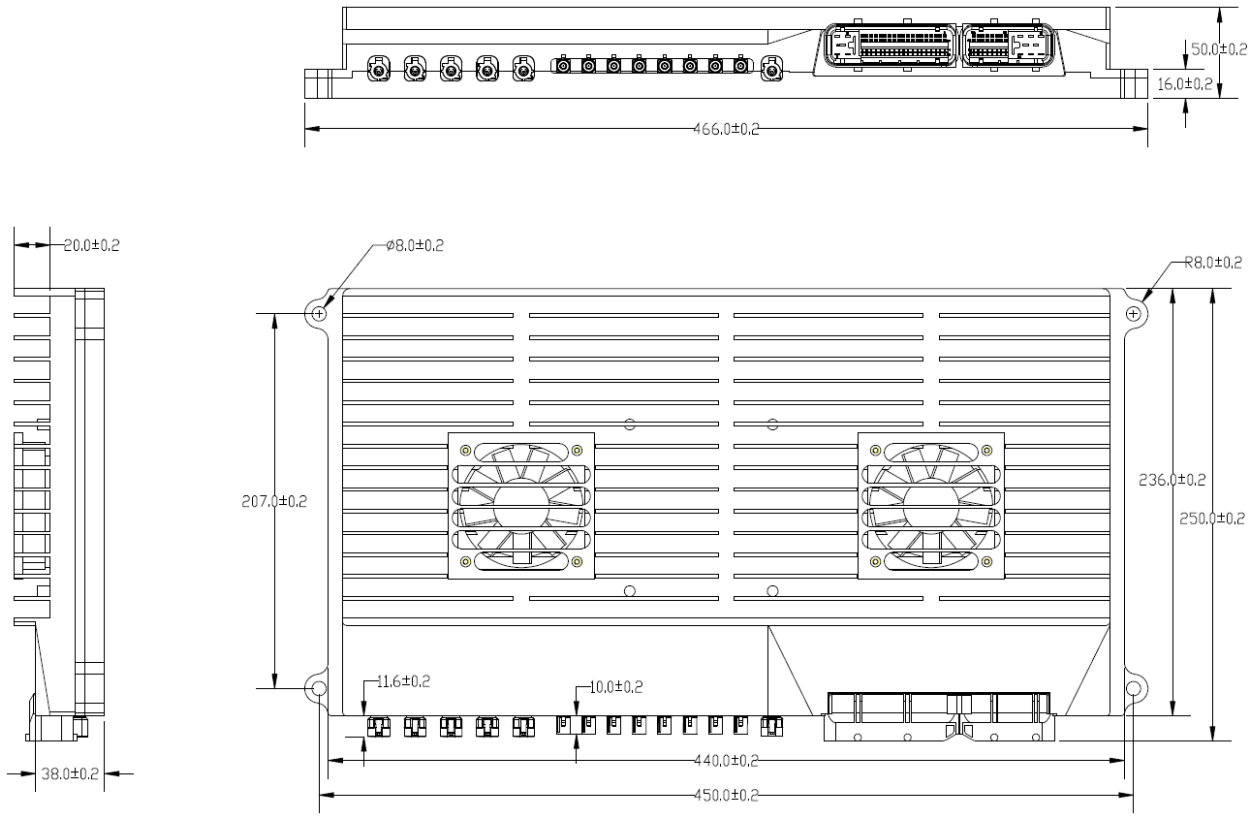
EAXVA03 is a central computing platform developed by Ecotrons LLC for autonomous driving systems using NVIDIA Jetson AGX Xavier and Infineon TC297, which supports up to ASIL-D functional safety. Developers can build a Level-4 autonomous driving system with EAXVA03 safely, conveniently and efficiently.

NVIDIA Jetson AGX Xavier is designed for embedded intelligent systems including autonomous driving systems. With over 9 billion transistors, it can perform 16 trillion floating-point operations per second only consuming 30W power, 12 times faster than the previous generation TX2 platform. Six different processors of Xavier enable it to process dozens of algorithms simultaneously and in real-time for sensor data processing, ranging, localization, mapping, visualization, perception, and path planning. The software system of Xavier is also customized for autonomous driving systems, which includes a real-time optimized Linux and a high-performance runtime framework ROS.

Infineon TC297 includes a TriCore™ architecture with a 300MHz operating frequency and an ECC (Error Correction Code) protected RAM with 728KB + 2MB capacity. It is designed based on the ISO26262 standard and supports up to ASIL-D functional safety. The software architecture is designed in strict accordance with AUTOSAR, divided into application software layer and basic software layer. The basic software layer is comprised of a microcontroller abstraction layer, an ECU abstraction layer, a service layer, and a complex driver. The application software layer and basic software layer are connected and integrated through the runtime environment. Developers can develop vehicle control strategies and functional safety based on the MCU.

Chapter 2 Mechanics

2.1 Dimensions



2.2 Connector

EAXVA03 uses the connectors from Tyco.

#	Connector	Name	Type	Supplier	Link
1	121P	PCB needle	1241434-1	TE	--
2		81P sheath	1473244-1	TE	http://www.digikey.com/products/en?keywords=1473244-1
3		40P sheath	1473252-1	TE	http://www.digikey.com/products/en?keywords=1473252-1
4		Terminal	964282-2	TE	http://www.digikey.com/products/en?keywords=964282-2
5		Terminal	968220-1	TE	http://www.digikey.com/products/en?keywords=968220-1
6		81P back	1473247-1	TE	http://www.digikey.com/products/en?keywords=1473247-1

7		40P back	1473255-1	TE	http://www.digikey.com/products/en?keywords=1473255-1
8		81P retainer	368382-1	TE	http://www.digikey.com/products/en?keywords=368382-1
9		40P retainer	368388-1	TE	http://www.digikey.com/products/en?keywords=368388-1
10	FAKRA	FAKRA needle	2291392-4	TE	https://www.te.com.cn/chn-zh/product-2291392-4.html
11	HSD	HSD needle	2291362-2	TE	https://www.te.com.cn/chn-zh/product-2291362-2.html
12		Harness connector	2282159-3	TE	https://www.te.com.cn/chn-zh/product-2282159-3.html

2.3 Mechanical Parameters

Dimensions: 466 × 250 × 50mm

Mechanical structure material: aluminum

Connectors: water-proof 121P connector, FAKRA interface, HSD interface

Housing rigidity: well

Chapter 3 Quick Start

3.1 Scenario One: Using EAXVA03 Alone

If you want to use EAXVA03 alone, i.e. without a development host, please prepare the following items:

Item	Notes
EAXVA03 with two ACU harnesses (81P and 40P)	
Stable DC power supply: 12V/5A	At least a 50W power supply is recommended.
Ethernet cable with RJ-45 connector	If you need connect to the internet.
Monitor	HDMI Interface needed.
HDMI harness	Connecting Monitor and EAXVA03
Mouse	USB interface needed.
Keyboard	USB interface needed.

To use the EAXVA03 alone, please follow the instructions below:

1. Using the 81P and 40P harnesses to connect the ACU and DC power supply. Keep the power supply off.
2. Connect the mouse, the keyboard, the monitor with the ACU.
3. Connect the ethernet cable with ACU and an internet terminal. (Optional. If you need connect to the internet.)
4. Turn on the power supply.
5. The default password for the system is “nvidia”.

Now you are able to use EAXVA03.

Note:

1. If the maximum current of the ACU is not enough, you may see the ACU shut down after self-check during starting, in which case, you need to use a power supply with a larger maximum current.

2. If you connect some cameras to the ACU, the working current will increase because camera power is supplied by the ACU, indirectly supplied by the power supply.
3. If you just turn off the ACU, before turning on the ACU again, you should wait for 10 seconds.

3.2 Scenario Two: Using EAXVA03 with Development Host

If you want to use EAXVA03 with the development host, please prepare the following items:

Item	Notes
EAXVA03 with two ACU harnesses (81P and 40P)	
Stable DC power supply: 12V/5A	At least a 50W power supply is recommended.
Ethernet cable with RJ-45 connector	To connect host with ACU, or connect the ACU to the internet.
USB to RS-232 adapter	Connect the host with ACU.
A development host	Ubuntu 18.04 recommended.
Monitor and HDMI harness, mouse, keyboard	Optional, if you want to use the ACU in the same way as scenario one.

To use the EAXVA03 with a development host, we recommend you follow the steps in scenarios one to find out the IP address of the ACU. Then you may use the ACU with the development host in two ways: through RS-232, or through ethernet.

Through RS-232 (You can also use this way to find out the IP address of the ACU):

1. Using the 81P and 40P harnesses to connect the ACU and DC power supply. Keep the ACU powered off.
2. Make sure Minicom is installed on the development host.
3. Launch Minicom. Turn on the ACU.
4. The username and password are both “nvidia”.

Through ethernet (This is also the way when you are using external mode of Simulink model):

1. Using the 81P and 40P harnesses to connect the ACU and DC power supply. Keep the power supply off.
2. Connect the ethernet cable with ACU and the host.
3. Turn on the power supply.
4. In the terminal on the host, type in ssh nvidia@<ACU IP Address>
5. The default password for the system is “nvidia”.

Now you are able to use EAXVA03 with the development host.

Note:

1. If the maximum current of the ACU is not enough, you may see the ACU shut down after self-check during starting, in which case, you need to use a power supply with a larger maximum current.
2. If you connect some cameras to the ACU, the working current will increase because camera power is supplied by the ACU, indirectly supplied by the power supply.
3. If you just turn off the ACU, before turning on the ACU again, you should wait for 10 seconds.

Chapter 4 Hardware

The hardware circuit of the device is designed according to the requirements of the autonomous driving applications. The electrical parameters can meet automotive-grade standards. It has various data transmission interfaces, which is very useful in sensor fusion of the autonomous driving system. The main chip contains a variety of high-performance computing units being able to process both sequential and parallel computing in the autonomous driving system.

- Operating voltage: DC 9-36V
- Operation memory: 16GB
- Storage memory: 32GB, SSD extendable
- Calculation capability: 32TOPS(INT8) or 16TFLOPS(FP16)
- Data transmission interfaces:
    - 10 Automotive Ethernet
    - 5 Standard Ethernet
    - 8 FPD-Link III video input
    - 1 FPD-Link III video output
    - 1 HDMI video output
    - 3 USB host
    - 1 USB OTG
    - 2 CAN
    - 6 CANFD
    - 2 LIN
    - 1 FlexRay
    - 4 RS-232
    - 1 RS-485
- I/O Interfaces
    - 20 digital input: 10 high effective (2/10 support hardwire wakeup), 6 low effective, 4 PWM input
    - 13 analog input: 4 resistance type, 9 voltage type (3@36V, 6@5V)

- 18 low side output: 4/18 can be configured to PWM output
- 10 high side output: 2/10 can be configured to PWM output
- 5 sensor power supply(5V): 2@max 100mA, 3@max 50mA

4.1 Specifications

Item	Parameter
Operating voltage	DC 9-36V
Operation memory	16GB
Storage memory	32GB
Operating temperature	-25 to 85 ℃
Operating humidity	0 - 95%, no condensation
Storage temperature	-40 to 125 ℃
Dimensions	466×250×50mm
Weight	≤5000g

4.2 Appearance

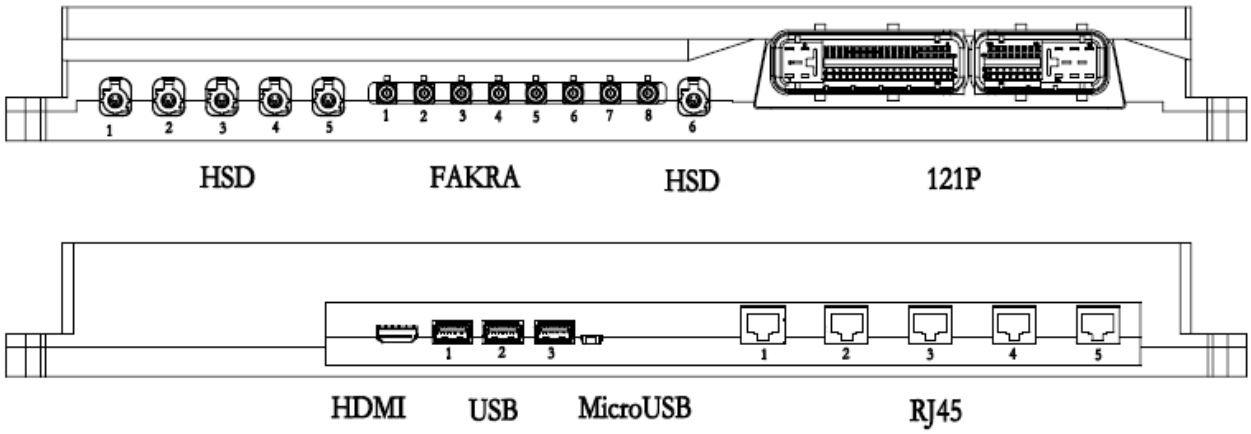
Vertical View

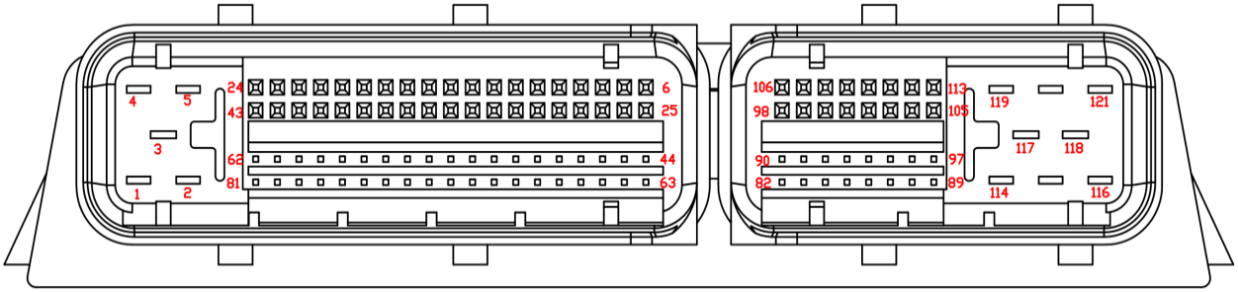
Front View

Rear View

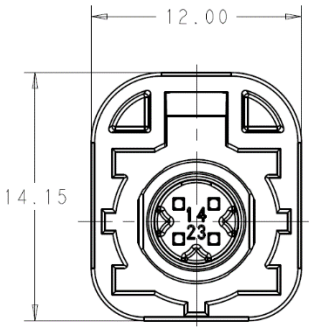
4.3 Device Ports

4.3.1 Port Placement





121P



HSD

4.3.2 Port Definition

Signal Name	PIN	Description	Note
Automotive Ethernet			
ENet1_N	HSD1-1	Automotive Ethernet 1	1000Base-T1 can be configured to 100Base-T1
ENet1_P	HSD1-2		
ENet2_N	HSD1-3	Automotive Ethernet 2	1000Base-T1 can be configured to 100Base-T1
ENet2_P	HSD1-4		
ENet3_N	HSD2-1	Automotive Ethernet 3	100Base-T1
ENet3_P	HSD2-2		
ENet4_N	HSD2-3	Automotive Ethernet 4	100Base-T1
ENet4_P	HSD2-4		
ENet5_N	HSD3-1	Automotive Ethernet 5	1000Base-T1 can be configured to 100Base-T1
ENet5_P	HSD3-2		
ENet6_N	HSD3-3	Automotive Ethernet 6	1000Base-T1 can be configured to 100Base-T1
ENet6_P	HSD3-4		

ENet7_N	HSD4-1	Automotive Ethernet 7	100Base-T1
ENet7_P	HSD4-2		
ENet8_N	HSD4-3	Automotive Ethernet 8	1000Base-T1 can be configured to 100Base-T1
ENet8_P	HSD4-4		
ENet9_N	HSD5-1	Automotive Ethernet 9	100Base-T1
ENet9_P	HSD5-2		
ENet10_N	HSD5-3	Automotive Ethernet 10	1000Base-T1 can be configured to 100Base-T1
ENet10_P	HSD5-4		
Camera			
Camera-1	FAKRA-1	FPD-Link III Serial Camera Interface 1	
Camera-2	FAKRA-2	FPD-Link III Serial Camera Interface 2	
Camera-3	FAKRA-3	FPD-Link III Serial Camera Interface 3	
Camera-4	FAKRA-4	FPD-Link III Serial Camera Interface 4	
Camera-5	FAKRA-5	FPD-Link III Serial Camera Interface 5	
Camera-6	FAKRA-6	FPD-Link III Serial Camera Interface 6	
Camera-7	FAKRA-7	FPD-Link III Serial Camera Interface 7	
Camera-8	FAKRA-8	FPD-Link III Serial Camera Interface 8	
Monitor			
DOUTA_P	HSD6-1	FPD-Link III Serial Display Interface 1	Display the same data
DOUTA_N	HSD6-2		
DOUTB_P	HSD6-3	FPD-Link III Serial Display Interface 2	
DOUTB_N	HSD6-4		
HDMI	HDMI	HDMI output interface	
USB			
USB Host-1	USB-1	USB interface 1	Support USB2.0, USB3.0, USB3.1
USB Host-2	USB-2	USB interface 2	Support USB2.0, USB3.0, USB3.1
USB Host-3	USB-3	USB interface 3	Support USB2.0, USB3.0, USB3.1
USB OTG	Micro USB	Micro USB interface	Support USB2.0 OTG
Standard Ethernet			
NPort-1	RJ45-1	Ethernet 1	Adaptive 1000Base-T/100Base-TX

NPort-2	RJ45-2	Ethernet 2	Adaptive 1000Base-T/100Base-TX
NPort-3	RJ45-3	Ethernet 3	Adaptive 1000Base-T/100Base-TX
NPort-4	RJ45-4	Ethernet 4	Adaptive 1000Base-T/100Base-TX
NPort-5	RJ45-5	Ethernet 5	Adaptive 1000Base-T/100Base-TX
Power			
BATT	121P-1	Power Positive	
	121P-3		
	121P-116		
	121P-118		
	121P-121		
Power Ground			
PGND	121P-2	Power Ground	
	121P-4		
	121P-5		
	121P-117		
	121P-119		
	121P-120		
Signal Ground			
GND	121P-36	Signal ground	Ground for 5V sensor power supply
	121P-45		
	121P-63		
	121P-65		
	121P-87		
Sensor Power Supply (5V)			
5V-1	121P-83	5V-1 Sensor Power Supply	Max current: 100mA
5V-2	121P-86	5V-2 Sensor Power Supply	Max current: 100mA
5V-3	121P-82	5V-3 Sensor Power Supply	Max current: 50mA
5V-4	121P-84	5V-4 Sensor Power Supply	Max current: 50mA
5V-5	121P-85	5V-5 Sensor Power Supply	Max current: 50mA
Wakeup Signal			
KEYON1	121P-44	KEYON1	

KEYON_56	121P-56	KEYON_56	High effective, hardwire wakeup signal
DC WAKE	121P-76	DC_WAKE	High effective, hardwire wakeup signal
Analog Input			
AI01	121P-42	Analog Input 0~5V (Voltage type)	12-bit resolution
AI02	121P-60	Analog Input 0~5V (Voltage type)	12-bit resolution
AI03	121P-43	Analog Input 0~5V (Voltage type)	12-bit resolution
AI04	121P-24	Analog Input 0~5V (Voltage type)	12-bit resolution
AI05	121P-79	Analog Input 0~5V (Voltage type)	12-bit resolution
AI06	121P-23	Analog Input 0~5V (Voltage type)	12-bit resolution
AI09	121P-59	Analog Input 0~5V (resistance type)	12-bit resolution
AI10	121P-80	Analog Input 0~5V (resistance type)	12-bit resolution
AI11	121P-81	Analog Input 0~5V (resistance type)	12-bit resolution
AI12	121P-78	Analog Input 0~5V (resistance type)	12-bit resolution
AI13	121P-62	Analog Input 0~36V (Voltage type)	12-bit resolution
AI14	121P-40	Analog Input 0~36V (Voltage type)	12-bit resolution
AI15	121P-22	Analog Input 0~36V (Voltage type)	12-bit resolution
Digital Input			
DI01	121P-20	Digital Input 0~BATT	High effective
DI02	121P-58	Digital Input 0~BATT	High effective
DI03	121P-77	Digital Input 0~BATT	High effective
DI04	121P-38	Digital Input 0~BATT	High effective
DI07	121P-15	Digital Input 0~BATT	High effective
DI08	121P-72	Digital Input 0~BATT	High effective
DI09	121P-21	Digital Input 0~BATT	High effective
DI10	121P-39	Digital Input 0~BATT	High effective
DI13	121P-18	Digital Input 0~BATT	Low effective
DI14	121P-75	Digital Input 0~BATT	Low effective
DI15	121P-57	Digital Input 0~BATT	Low effective
DI16	121P-37	Digital Input 0~BATT	Low effective

DI19	121P-35	Digital Input 0~BATT	Low effective
DI20	121P-17	Digital Input 0~BATT	Low effective
DI21	121P-74	Digital Input 0~BATT	High effective, can be configured to PWM input
DI22	121P-16	Digital Input 0~BATT	High effective, can be configured to PWM input
DI23	121P-73	Digital Input 0~BATT	Low effective, can be configured to PWM input
DI24	121P-19	Digital Input 0~BATT	Low effective, can be configured to PWM input
Output Signal			
HSO01	121P-88	Typical 1A，Maximum 3A	Can be configured to PWM output
HSO02	121P-89	Typical 1A，Maximum 3A	Can be configured to PWM output
HSO03	121P-97	Typical 1A，Maximum 3A	
HSO04	121P-96	Typical 0.5A，Maximum 1A	
HSO05	121P-104	Typical 0.5A，Maximum 1A	
HSO06	121P-105	Typical 0.5A，Maximum 1A	
HSO07	121P-113	Typical 1A，Maximum 3A	
HSO08	121P-112	Typical 1A，Maximum 3A	
HSO09	121P-115	Typical 3A，Maximum 5A	
HSO10	121P-114	Typical 3A，Maximum 5A	
LSO01	121P-101	Typical 0.5A，Maximum 1A	Can be configured to PWM output
LSO02	121P-94	Typical 0.5A，Maximum 1A	Can be configured to PWM output
LSO03	121P-90	Typical 0.5A，Maximum 1A	Can be configured to PWM output
LSO04	121P-92	Typical 0.5A，Maximum 1A	Can be configured to PWM output
LSO05	121P-110	Typical 0.5A，Maximum 1A	
LSO06	121P-103	Typical 0.5A，Maximum 1A	
LSO07	121P-109	Typical 0.5A，Maximum 1A	
LSO08	121P-107	Typical 0.5A，Maximum 1A	
LSO09	121P-100	Typical 0.5A，Maximum 1A	

LSO10	121P-102	Typical 0.5A，Maximum 1A	
LSO11	121P-91	Typical 0.5A，Maximum 1A	
LSO12	121P-93	Typical 1A，Maximum 3A	
LSO13	121P-111	Typical 1A，Maximum 3A	
LSO14	121P-95	Typical 1A，Maximum 3A	
LSO15	121P-108	Typical 1A，Maximum 3A	
LSO16	121P-99	Typical 1A，Maximum 3A	
LSO17	121P-98	Typical 3A，Maximum 5A	
LSO18	121P-106	Typical 3A，Maximum 5A	
Communication Port			
CANA_H	121P-31	With 120 Ω Terminal Resistor	
CANA_L	121P-32		
CANB_H	121P-11	With 120 Ω Terminal Resistor	
CANB_L	121P-12		
CANC_H	121P-29	With 120 Ω Terminal Resistor	
CANC_L	121P-30		
CAND_H	121P-13	With 120 Ω Terminal Resistor	
CAND_L	121P-14		
CANE_H	121P-27	Without 120 Ω Terminal Resistor	Support wakeup by user-defined message ID
CANE_L	121P-28		
CANF_H	121P-9	Without 120 Ω Terminal Resistor	Support wakeup by user-defined message ID
CANF_L	121P-10		
CANG_H	121P-47	With 120 Ω Terminal Resistor	Xavier: CAN0
CANG_L	121P-66		
CANH_H	121P-48	With 120 Ω Terminal Resistor	Xavier: CAN1
CANH_L	121P-67		
CAN_SHILD-1	121P-46	CAN Shield	
CAN_SHILD-2	121P-8	CAN Shield	
LIN1	121P-7	LIN bus 1	Support wakeup
LIN2	121P-26	LIN bus 2	Support wakeup

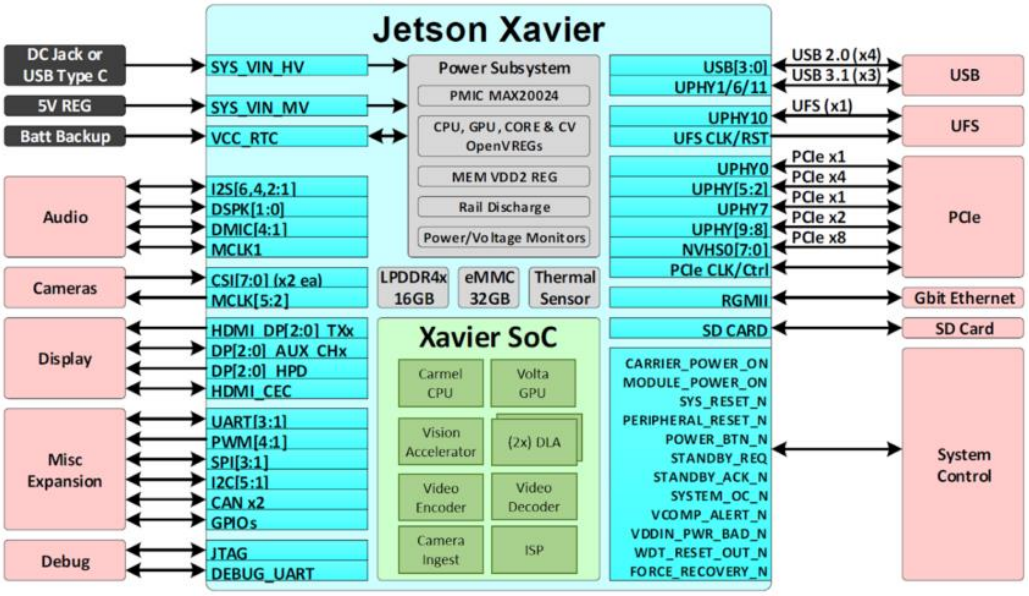
FLX_H	121P-25	FlexRay interface	Support wakeup
FLX_L	121P-6		
TXD1	121P-52	RS-232 interface 1	
RXD1	121P-71		
TXD2	121P-69	RS-232 interface 2	
RXD2	121P-50		
TXD3	121P-51	RS-232 interface 3	For debug
RXD3	121P-70		
TXD7	121P-41	RS-232 interface 7	
RXD7	121P-61		
Others			
PPS	121P-64	Pulse Per Second Synchronization Signal	Voltage = 3.3V
A	121P-34	RS485_A	
B	121P-33	RS485_B	

4.4 System Main Chip

The main chip of EAXVA03 is NVIDIA Jetson AGX Xavier which is designed for embedded intelligent systems including autonomous driving systems. Xavier possesses six different processors: Valta architecture GPU, 8-core Carmel ARM64 CPU, Deep Learning accelerator, Vision Accelerator, Video Encoder and Video Decoder. These processors enable dozens of algorithms to be processed simultaneously and in real-time for sensor data processing, ranging, localization, mapping, visualization, perception, and path planning. Powerful computation capability allows autonomous vehicles to take input from sensors, locate themselves, interpret the ambient environment, identify and predict the motion of nearby objects, act correspondingly and safely. Xavier can perform 32 TeraOPS (TOPS) with a power of only 30 watts, which is 10 times faster than Jetson TX2. The computational resources of different processors are as follows.

- CPU: 8-Core Carmel ARM v8.2 64-Bit CPU, 8 MB L2 + 4 MB L3
- Deep Learning Accelerator (DLA): 5 TFLOPS (FP16) | 10 TOPS (INT8)

- GPU: NVIDIA Volta™ architecture with 512 NVIDIA CUDA cores and 64 Tensor cores, 11 TFLOPS (FP16), 22 TOPS (INT8)
- Vision Accelerator: 7-Way VLIW Vision Processor
- Video Encoder: 2x1000 MP/sec
- Video Decoder: 2x1500 MP/sec



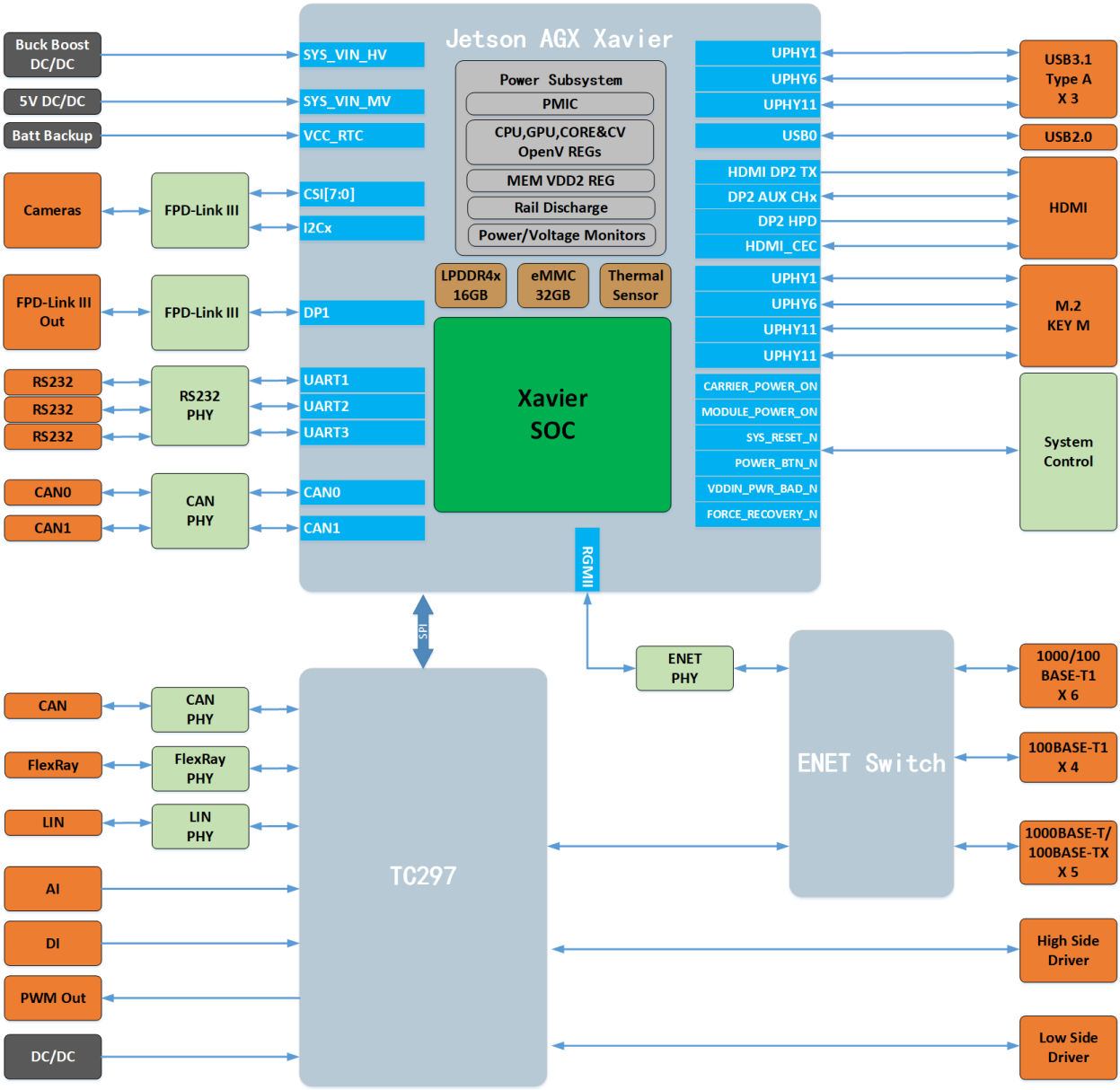
The microcontroller of EAXVA03 is Infineon TC297 which has a TriCore™ architecture working at 300MHz and an ECC (Error Correction Code) protected RAM with a capacity of up to 728KB + 2MB, designed based on ISO26262, supporting up to ASIL-D functional safety. Working with a basic chip, a hardware core security architecture design is realized. The resources of the chip are as follows:

Feature	**Detail**
Micro Control Core	32-bit Infineon TC297TP
Maximum Frequency	300MHz
Flash	8M
SRAM	728K

EEPROM	128K
Float Point Capability	Yes
SBC Microprocessor	TLE7368-3E

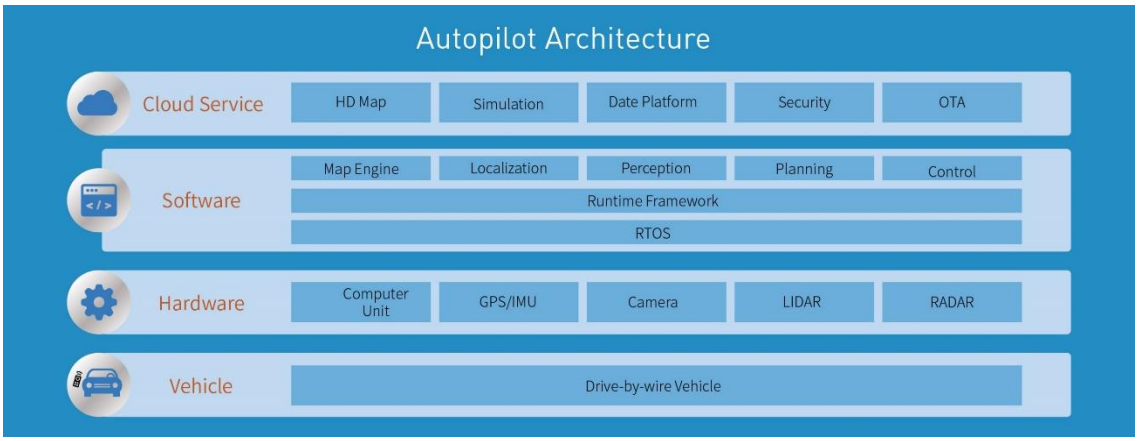
4.5 Circuit Structure

The internal circuit structure of EAXVA03 is shown below:



Chapter 5 SoC Basic Software

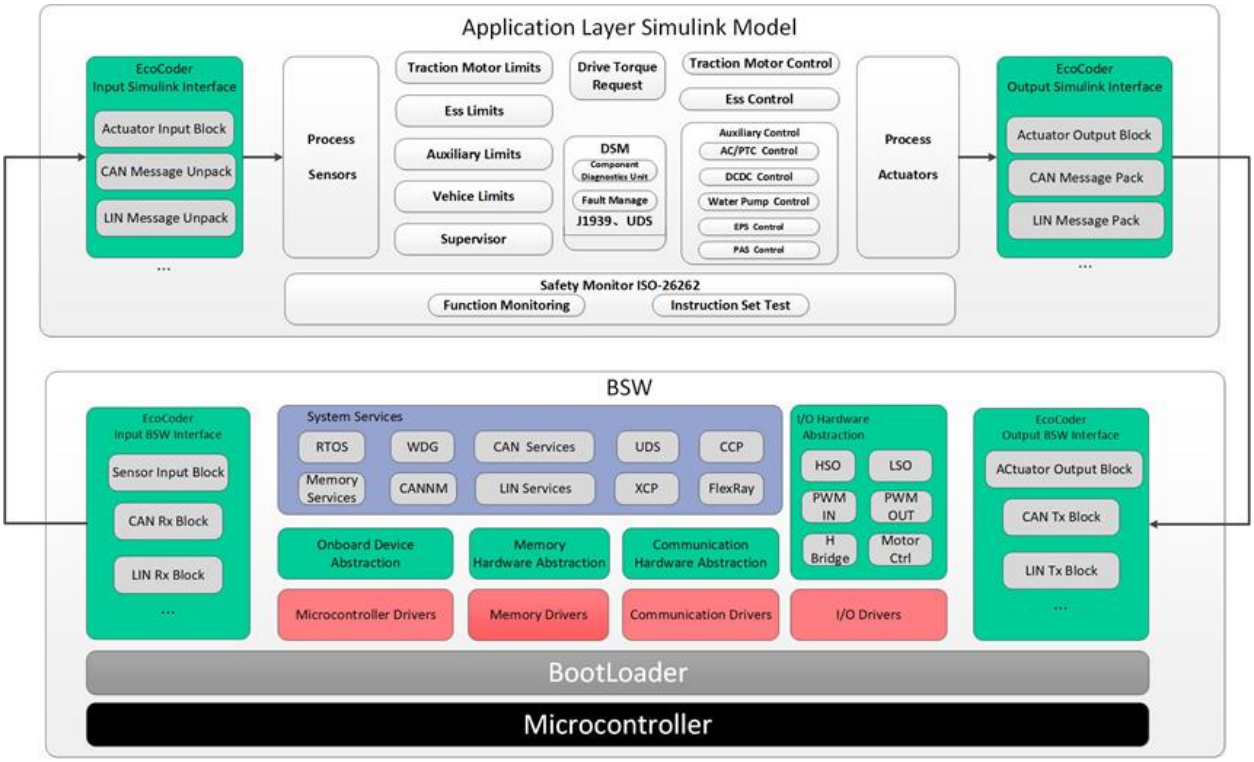
A typical framework of an autonomous driving system is shown below. The SoC software system of EAXVA03 consists of RTOS and Runtime Framework. RTOS is a Linux operating system. The Runtime Framework is ROS(Robot Operating System) Melodic. Linux is a bridge connecting the hardware and the users, providing functions such as Process Management, Memory Management, File System, Network, Security, User Interface, and Device Drivers. ROS provides some standard operating system services, such as Hardware Abstraction, Low-Level Device Control, Inter-Process Messaging, and Message Packet Management. ROS is built on a graph architecture, various nodes can publish, subscribe and aggregate all kinds of information, e.g. sensing, control, status, planning.



Chapter 6 MCU Basic Software

The software architecture of the MCU inside EAXVA03 is designed according to AUTOSAR, which is divided into Application Software Layer and Basic Software Layer. Basic Software Layer consists of a microcontroller abstraction layer, an ECU abstraction layer, a service layer, and a complex driver. Application software and basic software are connected and integrated through EcoCoder. EcoCoder encapsulates the low-level software interfaces into the Simulink library via s-functions. Application developers can use Simulink to build the model and generate executable program files for TC297 via Simulink by just one click.

The low-level software interfaces that EcoCoder encapsulates can read digital and analog input signals, control high and low side outputs, support .dbc file interpretation, implement CCP and UDS protocols, and define the measurement, calibration and NVM variables. MCU application development is implemented with the calibration software EcoCAL and the flashing software EcoFlash.



Chapter 7 Interface

In a Linux system, all devices are divided into three categories: character devices, block devices, and network interfaces.

For EAXVA03, different interfaces correspond to different types of devices in the Linux system. RS232 and camera belong to the character device, Ethernet and CAN belong to the network interface. For these devices, users can use C program to call driver access functions to access the interfaces. It's shown below how to configure and access the interfaces through terminal and how to access the interfaces using C program.

7.1 RS232

The RS232 interface is mapped to character device in Linux in EAXVA03. RS232-1, RS232-2 and RS232-3, and RS232-7 correspond to /dev/ttyTHS0, /dev/ttyTHS1, /dev/ttyTCU0 and /dev/ttyTHS3 respectively.

To view the parameters of RS232, use this command:

```
# stty -F /dev/ttyTHS0 -a
```

To set the baud rate of RS232-1(/dev/ttyTHS0) as 115200 and 8-bit data mode, use command below. If the data can't display properly, you may use command `stty --help` to see other configuration options.

```
# stty -F /dev/ttyTHS0 ispeed 115200 ospeed 115200 cs8
```

To print the data from RS232 on Terminal, use this command:

```
# cat /dev/ttyTHS0
```

To send data by RS232, you can use this command:

```
# echo "hello world" > /dev/ttyTHS0
```

For more info about the message transceive of RS232 in C programming language, please refer to Serial Programming Guide for POSIX Operating Systems.

7.2 CAN

In EAXVA03, CAN interface is mapped to SocketCAN in Linux, which is managed as a network device by the system.

To view CAN devices, you can use the command below. As shown below, eth0 is ethernet interface, can0 and can1 are CANG and CANH respectively.

```
# ifconfig –a
```

```
can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

can1      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:30
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:03:00:00:02:41
          inet addr:192.168.1.238  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:146 errors:0 dropped:0 overruns:0 frame:0
          TX packets:85 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10061 (9.8 KiB)  TX bytes:5447 (5.3 KiB)
          Interrupt:21 Base address:0x4000
```

To set the baud rate of CAN interface as 500Kbps:

```
# ip link set can0 type can bitrate 500000
```

To view the parameters of CAN interface:

```
# ip -details link show can0
```

To enable a CAN interface:

```
# ifconfig can0 up
```

To disable a CAN interface:

```
# ifconfig can0 down
```

To send a CAN frame with ID 0x5A0 and data 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88:

```
# cansend can0 -i 0x5a0 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
```

To receive data from CAN and display it on the terminal:

```
# candump can0
```

For more info about how to transceive CAN messages by SocketCAN in C programming language, please refer to SocketCAN Documentation (Linux Kernel). To know more about SocketCAN-based APPs in Linux, please refer to linux-can • GitHub.

7.3 Ethernet

In EAXVA03, the ethernet interface is mapped to eth0 in Linux, which is managed as a network device by the system.

To view the Ethernet interface, use the command below. As shown below, eth0 is an Ethernet interface.

```
# ifconfig –a
```

```
can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

can1      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:30
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:03:00:00:02:41
          inet addr:192.168.1.238  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:146 errors:0 dropped:0 overruns:0 frame:0
          TX packets:85 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10061 (9.8 KiB)  TX bytes:5447 (5.3 KiB)
          Interrupt:21 Base address:0x4000
```

To set the IP address to be obtained automatically, you can use the vi editor to open the network port configuration file and edit the file as follows, then restart the network port device. If you need help with how to use vi editor, please refer to A Beginner's Guide to Vim.

```
# vi /etc/network/interfaces
```

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

# The loopback interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
dhclient_opts -n

auto can0
iface can0 inet manual
        pre-up /sbin/ip link set $IFACE type can bitrate 500000
        up /sbin/ifconfig $IFACE up
        down /sbin/ifconfig $IFACE down

auto can1
iface can1 inet manual
        pre-up /sbin/ip link set $IFACE type can bitrate 500000
        up /sbin/ifconfig $IFACE up
        down /sbin/ifconfig $IFACE down
```

```
# /etc/init.d/networking restart
```

Or you can set the IP address fixed: use vi editor to open the Ethernet configuration and edit the file as follows, then restart the Ethernet device.

```
# vi /etc/network/interfaces
```

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

# The loopback interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 10.0.0.100
        netmask 255.255.255.0
        gateway 10.0.0.1

auto can0
iface can0 inet manual
        pre-up /sbin/ip link set $IFACE type can bitrate 500000
        up /sbin/ifconfig $IFACE up
        down /sbin/ifconfig $IFACE down

auto can1
iface can1 inet manual
        pre-up /sbin/ip link set $IFACE type can bitrate 500000
        up /sbin/ifconfig $IFACE up
        down /sbin/ifconfig $IFACE down
```

```
# /etc/init.d/networking restart
```

For more info about socket network programming in C programming language, please refer to Unix Socket Tutorial or Beej's Guide to Network Programming.

7.4 Camera

In EAXVA03, the camera interface is mapped to a standard video input device in Linux. Camera1, Camera2, … correspond to /dev/video0, /dev/video1, … respectively. There are two ways to access the image of the camera. Here, we will take /dev/video0 as an example:

1. Using nvgstcapture-1.0

```
# nvgstcapture-1.0 --cap-dev-node=0
```

--cap-dev-node is the video device node, 0=/dev/video0 by default, 1=/dev/video1 and so on.

2. Based on Argus

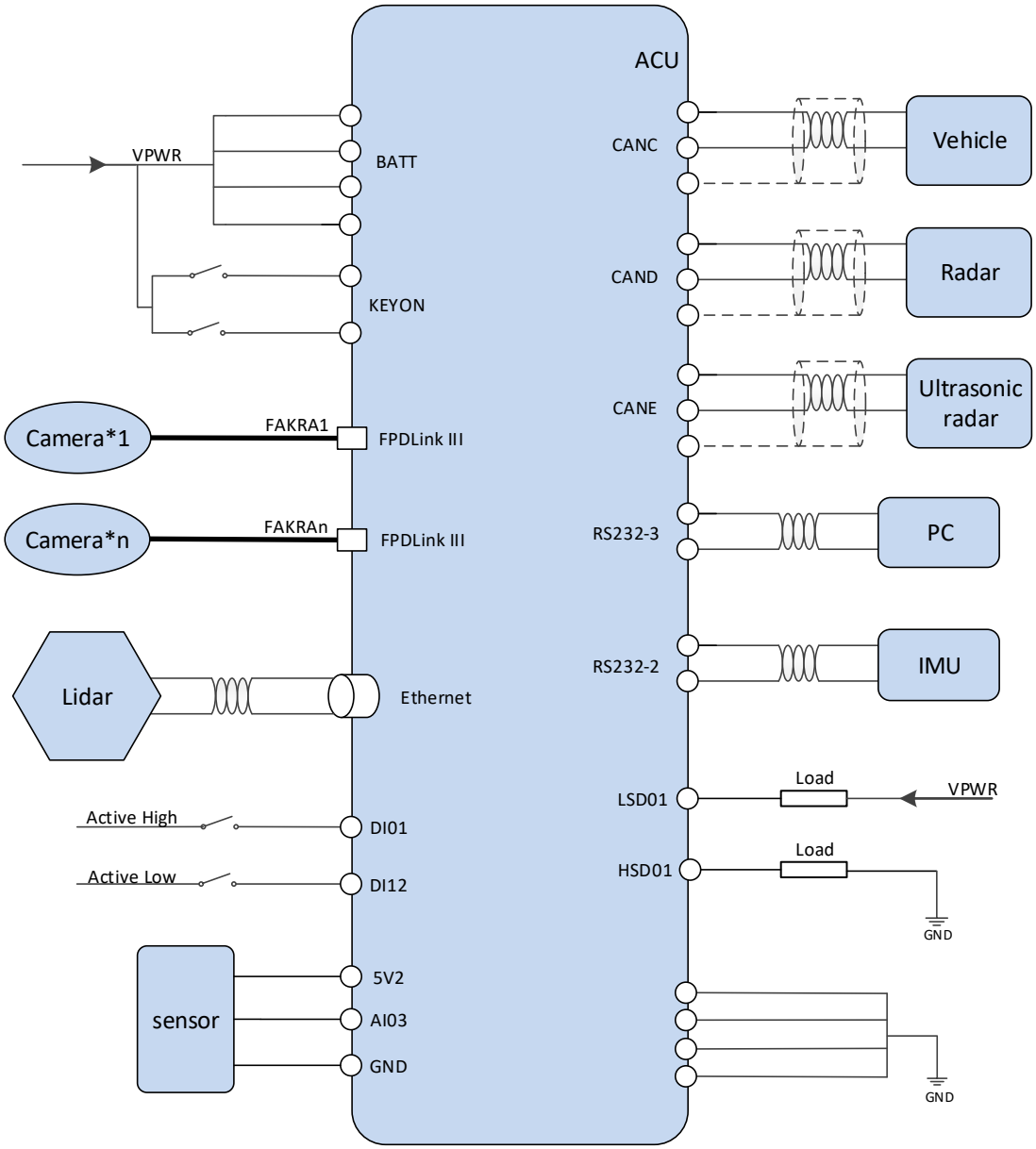
NVIDIA Multimedia API is pre-installed by default. To capture the image from Camera1:

```
# argus_camera --device=0
```

If you need to capture the image from the camera using C programming language, please refer to CSI-Camera.

Chapter 8 Demo Application

A demo for an autonomous driving hardware platform is shown below, which consists of EAXVA03 and sensors. Different cameras use different drivers, therefore, if you need EAXVA03 to work with a specific camera, please contact the camera supplier.



Chapter 9 Development Tool

A combination of hardware, operating system stacks, and runtime environments are not capable enough to realize autonomous driving, therefore, users need to develop software packages that can perform specific functionality and deploy them to EAXVA03. For autonomous driving processor Xavier, three development tools are provided: Local Development ToolKit, EcoSDK-XV, and EcoCoder-AV. For MCU Infineon TC297, three development tools are provided: EcoCoder, EcoCAL, and EcoFlash. Developers can select the tools they need.

9.1 Local Development ToolKit

EAXVA03 has pre-installed a set of local development tools, including gcc, make, CMake, catkin, Bazel and gdb debugger. Application developers can develop user-space applications directly on the EAXVA03 platform.

9.2 EcoSDK-XV

EcoSDK-XV provides users with a complete application development environment, including:

- Cross-development toolchain: consists of a cross-compiler, cross-connector, cross-debugger, and a set of other tools for application development.
- System root: EcoSDK-XV contains 2 system roots. One is for the development host, which contains the cross-development toolchain and other tools; the other is a complete root file system for the target, also contains development kits including header files and libraries.
- Environment configuration: The script provided by the EcoSDK-XV package allows users to configure an environment for cross-development on the development host.
- Analysis tools: userspace tools for analyzing applications on the target system.

EcoSDK-XV gives application developers all the tools necessary to write applications based on Linux, ROS, and Apollo Cyber RT. For details, please refer to EcoSDK-XV Manual.

9.3 EcoCoder-AV

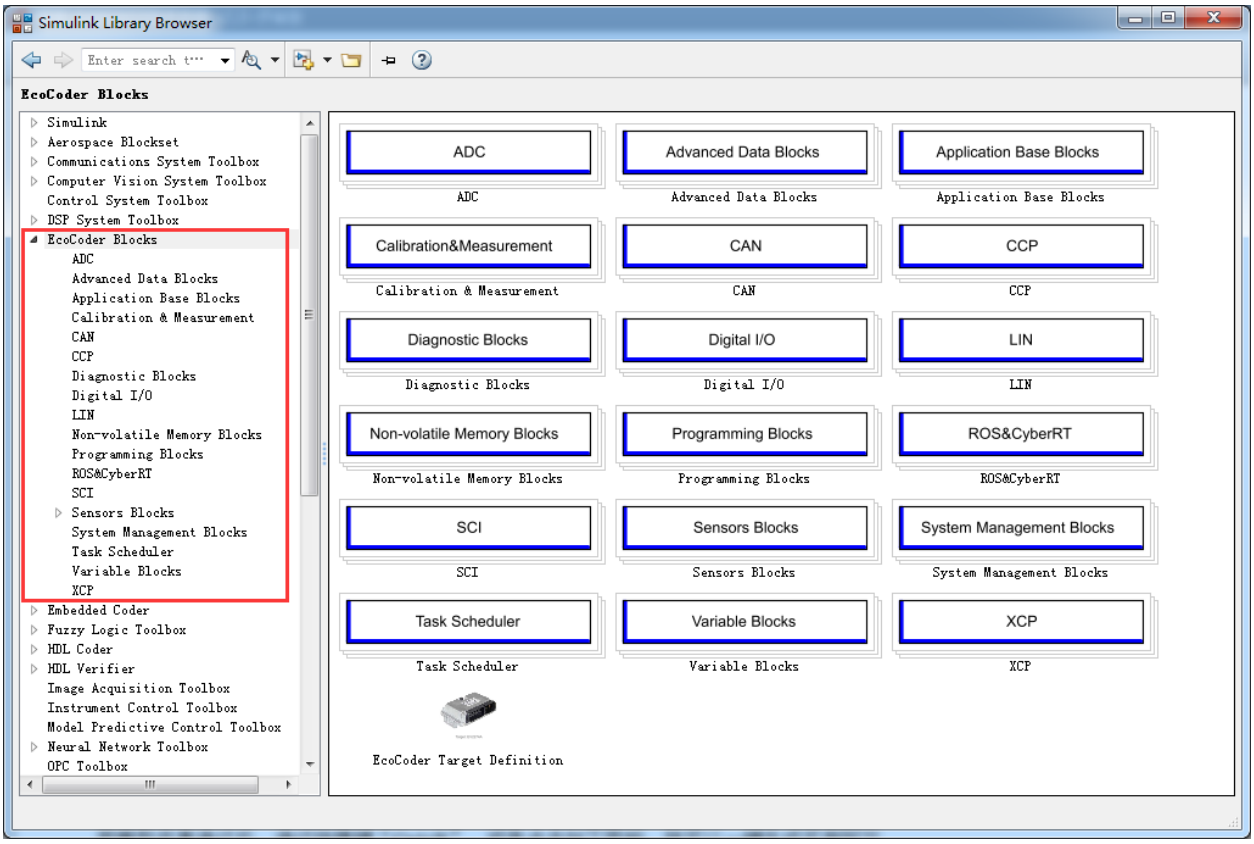
EcoCoder-AV is a powerful automatic code generation library based on Matlab / Simulink that links directly to the target controller. EcoCoder-AV integrates code generation, compilation and one-click generation of executable files. It can directly convert the Simulink model into an Apollo Cyber RT-based or ROS-based executable program for the target controller and download it to the target controller. In addition, it supports external mode which enables users to calibrate the model on-the-fly, which means users can change the variables of the model while the model is running on the target. For details, please refer to EcoCoder-AV Manual.

9.4 EcoCoder

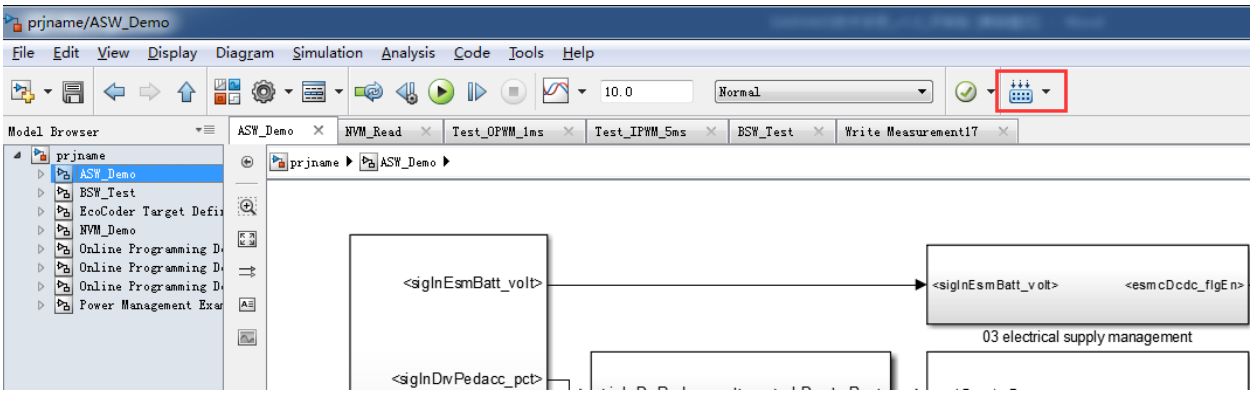
EcoCoder is an application development tool for the control system, which makes it easier for users to develop embedded application software in the Simulink environment. It expands the resources of Simulink and Real-Time Workshop embedded encoders to generate the necessary code module and automatically configures and optimizes code generation. By encapsulating the low-level software library to s-functions, EcoCoder allows developers to use low-level software interfaces by model-based-design method and configure basic parameters. It can generate executable files and data description files with one click and provide .a2l file address update tool.

Features:

- Users develop embedded application software in the Simulink environment.
- Application developers can focus on control strategy development without knowing all the information about hardware.
- By encapsulating the low-level software library to s-functions, EcoCoder enables developers to use the low-level software interfaces and configure parameters using the model-based-design method.
- Executable file and data description file can be generated by one click, and a .a2l file address update tool is provided. During the generation, the code generated by the model is integrated with the low-level software automatically in the background, then makefile is used to call the compiler to generate executables.



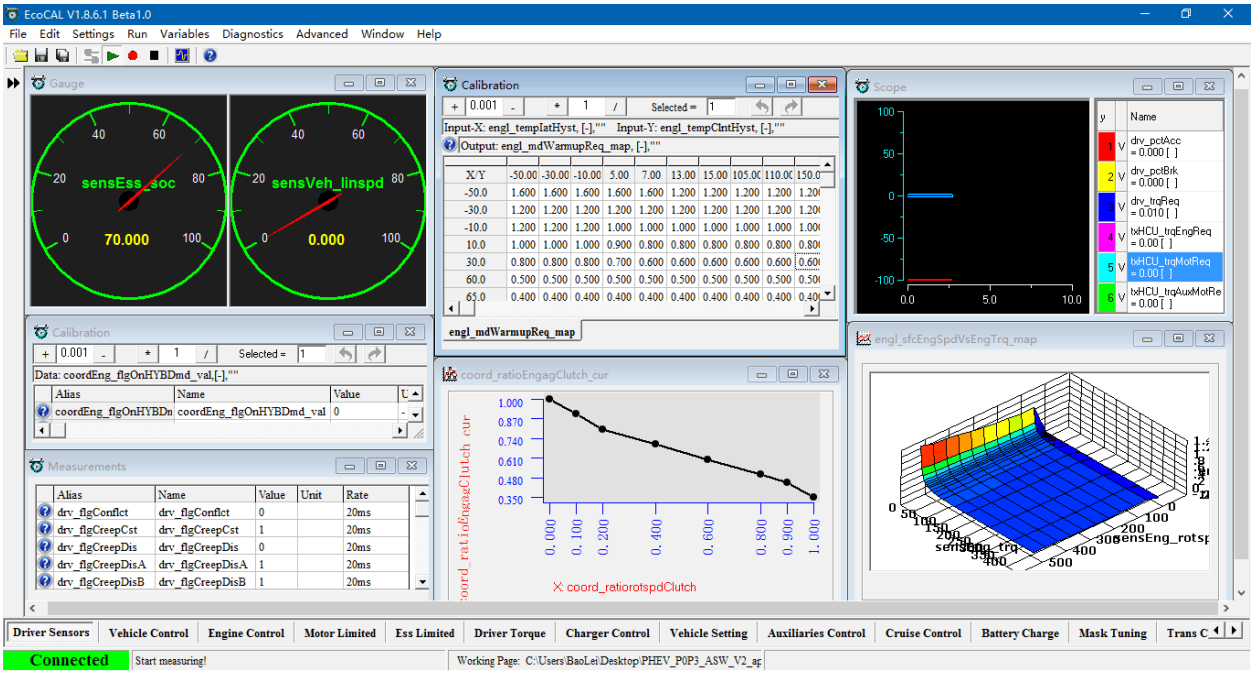
After compilation of the model, use the shortcut “Ctrl + B” or click the button shown below, the files ready to be flashed will be generated.



Developers can use EcoCoder to develop application software for MCU in EAXVA03. Please refer to EcoCoder User Manual.

9.5 EcoCAL

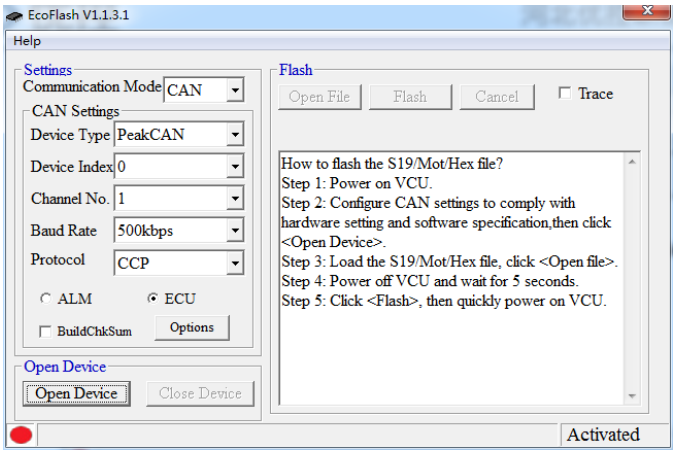
EcoCAL is a PC-side calibration software based on the CCP protocol. By loading .a2l and .hex files, real-time observation of the measurable variables and on-the-fly calibration can be realized. It can assist control strategy development engineers to debug and calibrate application software. Please refer to EcoCAL User Manual for more details.



9.6 EcoFlash

9.6.1 EcoFlash Introduction

EcoFlash is PC-side software working with BootLoader to flash target program files. The CAN communication uses CCP/UDS protocol, and .s19, .mot and .hex files are supported.



9.6.2 Activation Mode for Flash Failure

During the flash process using EcoFlash, make sure you are following the correct procedure, and do not power off or cut off the communication.

If the controller powers off during flashing unexpectedly, flashing might fail, wherein you cannot flash it again using non-default parameters, i.e. CRO, DTO and baud rate in the controller parameter segment are not default value.

If the problem above happens, you can use Rescue Mode to recover the controller:

1. If the CAN bus cuts off during flashing

The controller is not locked. You can reconnect the CAN bus and redo the flashing process in the correct way.

2. If the controller powers off during flashing

In this case, if you have changed the parameters(in an EcoCoder block called Online Programming, such as CRO, DTO, baud rate) before flashing, the controller might be locked and it won't be able to reflash again. If this happens, in the EcoFlash, you can set CRO = 0x100, DTO = 0x101, baud rate = 500kbps, and reflash it.

If the controller still cannot be restored or you have forgotten the flash parameters (CRO, DTO, etc.), Activation Mode is needed:

1) Connect DI01, DI02, DI03, and DI04 to 12V.

2) Connect DI13, DI14, DI15, and DI16 to GND.

3) Open EcoFlash and set the parameters: baud rate = 100kbps, CRO ID = 0x100, DTO ID = 0x101.

4) Flash the controller in the correct way, and Key Cycle is needed(Key Cycle: power off and power on. If the controller is powered off, you can power on directly.) Online flashing is not supported at this time.

5) The controller should be restored now. Now you can flash the controller using normal flashing procedure.