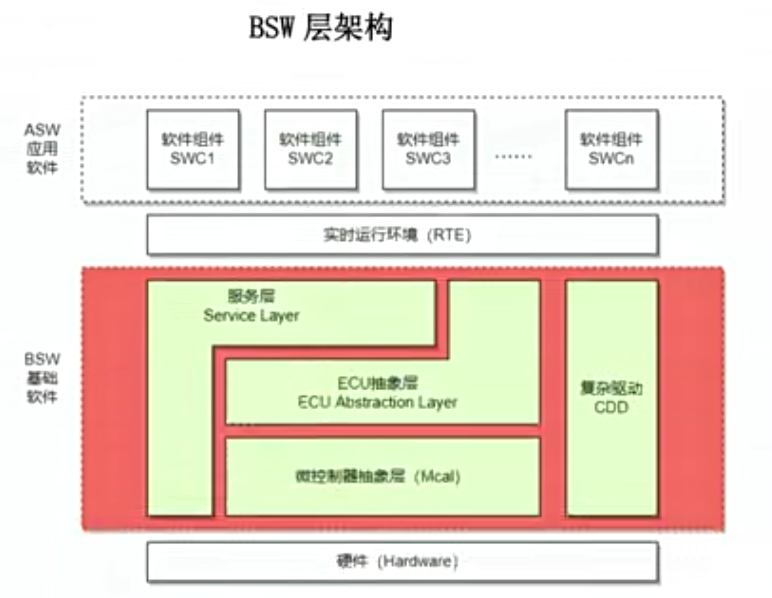
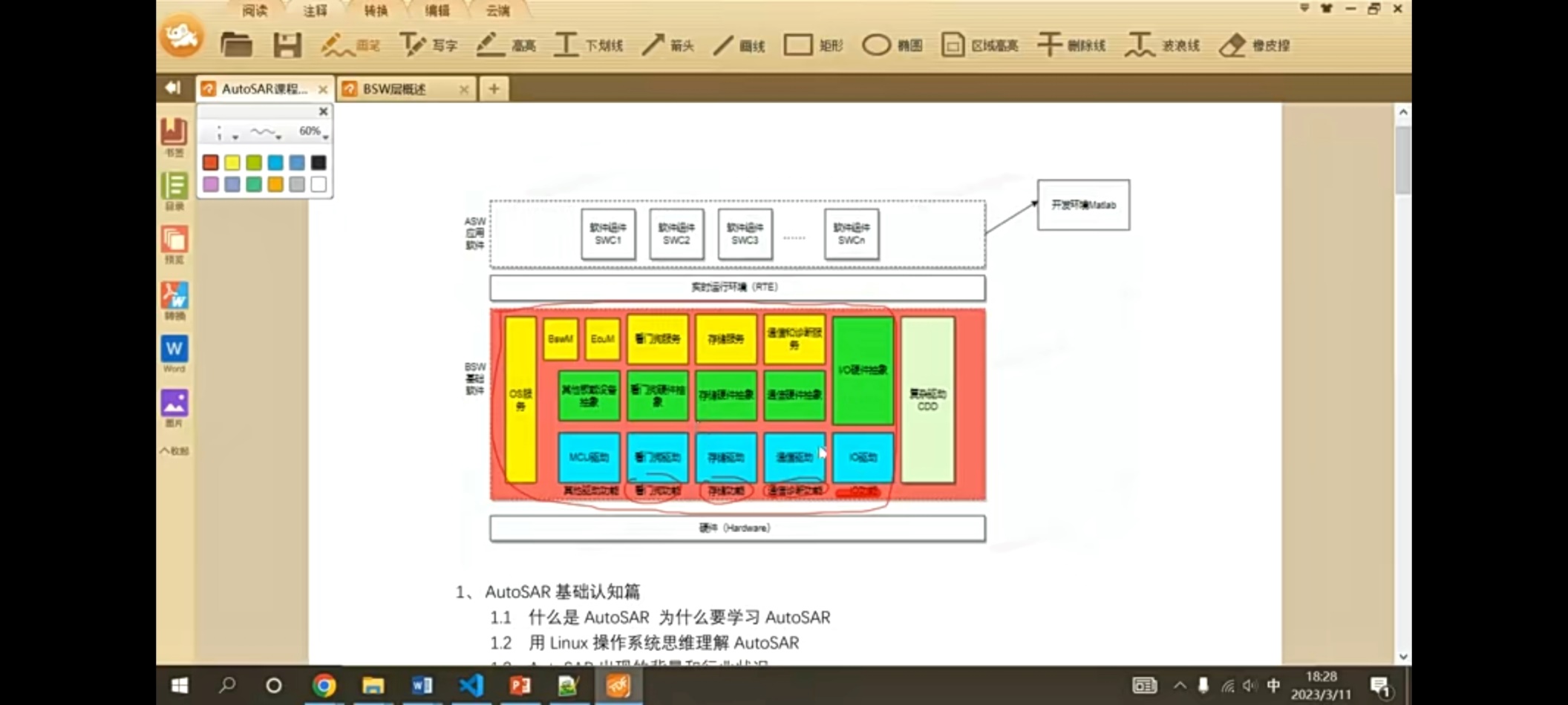
# 10\_AutoSAR技术讲解-10-BSW概述-1

Basic Software简称BSW。



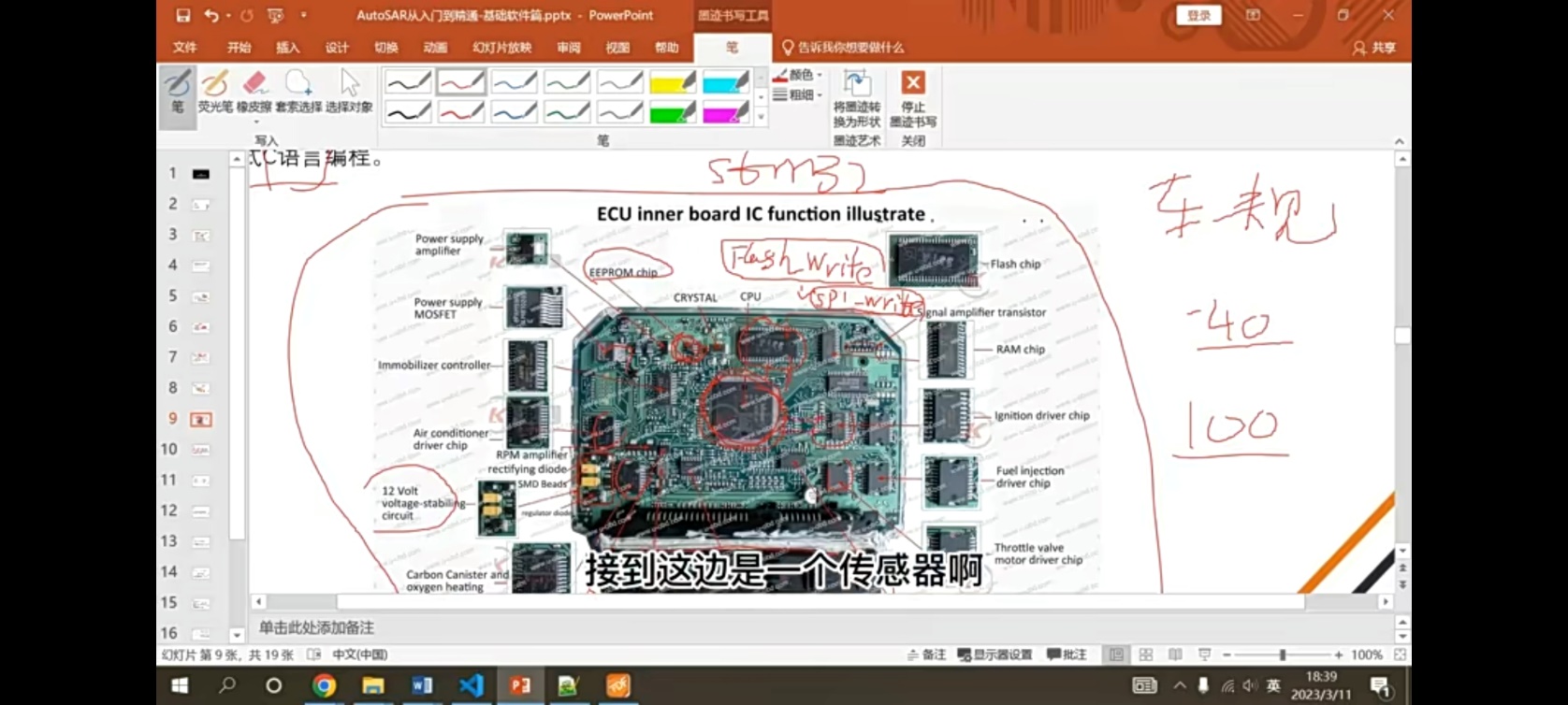
基础软件主要实现了芯片资源的管理，相当于操作系统底层实现的功能，即为应用层提供基础服务的。

基础软件标准分层3层，一个是mcal，Microcontroller 微控制器抽象层。微控制器即MCU，它把MCU芯片的各种外设给封装成标准的Autosar的接口，给到ECU抽象层使用。ECU抽象层其实就是各种协议栈，比如通信协议栈，存储协议栈。服务层类似操作系统OS，可以直接去驱动芯片的硬件，还有类似各种模式管理的一些模块，比如ECUM等。还有一块是复杂驱动，从上到下直接控制硬件，接口可以直接给到应用层使用。



复杂驱动不是指驱动很复杂，比如说有一些外设驱动，CAN总线的收发器，LIN总线的收发器，可能这些触发器的GPIO拉高一下就使能了。那么这种驱动就不好划分到Autosar的标准分层中去了，因为它就是一个IO口不好划分。像这样就可以直接写一个复杂驱动，应用层直接调一个GPIO的write接口就可以使能硬件的接口了。

通信，存储，看门狗，还有IO驱动就可以作为一个非常标准的Autosar的协议栈。驱动层又封了一层，ECU抽象层，还可以封一层系统服务层，比如IO硬件抽象就没有系统服务层，就可以直接给到应用层使用了。



芯片资源：

IO:GPIO ADC PWM

通信：CAN LIN ETH SPI

存储：FLASH EEPROM

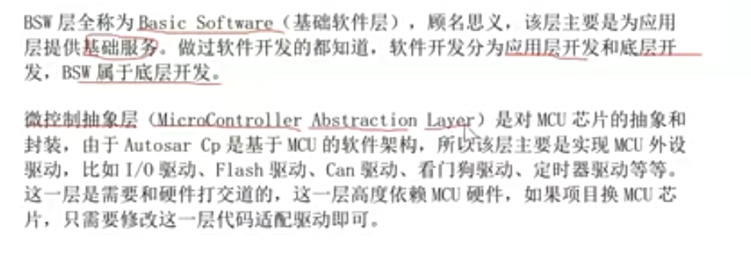
看门狗

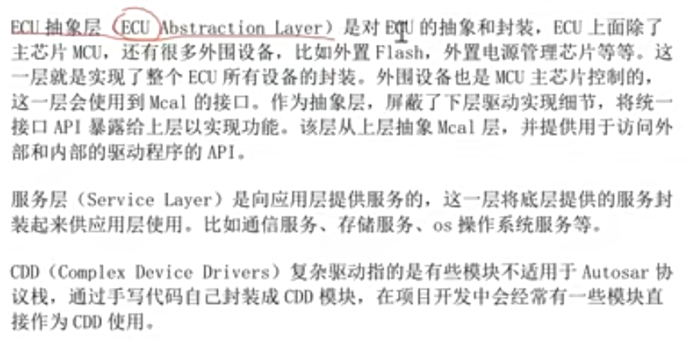
基础软件就是给应用层提供了接口去使用上述这些硬件资源。

软件开发分为应用层软件开发和底层软件开发，BSW属于底层软件开发，可以理解为做操作系统的底层。主要有驱动，协议栈，系统服务。微控制器抽象层 Microcontroller Abstraction Layer，根据Autosar的规范封装了微控制器MCU各种外设驱动，不仅仅是配寄存器，除了配置寄存器之外，还需要向上向Autosar提供标准的接口，它是对MCU芯片的抽象和封装。

由于Autosar Cp是基于MCU的软件架构，所以该层主要是实现MCU外设驱动，比如I/O驱动，Flash驱动，CAN驱动，看门狗驱动，定时器驱动等。这一层是需要和硬件打交道的，这一层高度依赖MCU硬件，如果项目换MCU芯片，只需要修改这一层代码适配驱动即可。

比如英飞凌的TC397，NXP的S32G，不同的芯片，MCAL层配置出来的代码时不一样的，但是当用397开发产品的时候，开发完了，后面需要替换芯片的时候。那么去修改Mcal层的代码就可以了，上面的代码不需要修改了，因为做了分层隔离了。应用层的代码就更不需要修改了，这既是Autosar实现标准分层的意义，可以最大程度的复用上层软件。





MCAL往上是ECU抽象层。ECU是电子控制单元，一块电路板，里面有MCU主芯片。这一层是对整个产品的封装。产品中除了主芯片还有很多外围设备，这里的外部设备不是指芯片外设。外设是指MCU内部的CAN控制器，SPI控制器。主芯片的外围设备有EEPROM，外置的flash，还有外置的电源管理芯片。这些外围设备会在ECU抽象层封装出外围设备使用接口。管理这些外围设备的这个接口，比如电源管理芯片需要通过SPI总线和主芯片进行通信控制的。所以主芯片的开发过程中最底层的MCAL驱动需要SPI的外设驱动，再往上ECU抽象层就可以封装成外置的Flash使能的一些接口。（MCAL就是封装MCU外设驱动，ECU抽象层就是封装ECU产品外围设备接口，ECU抽象层会使用MCAL驱动。）

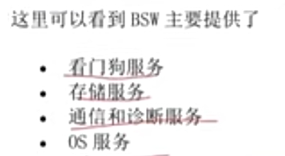
比如说叫，flash\_write(ECU抽象层接口)，这个函数往下调用的是SPI\_write。也就是说mcal使用的是spi\_write接口，mcal仅仅只是去实现spi MCU外设驱动，但是ECU抽象层会知道，SPI会使能flash。也有可能还有一路SPI，接到这边是一个传感器，然后ECU抽象层这一层可能就会有一个传感器的write接口，但是底层调用的驱动都是SPI\_write的驱动。只不过一路是SPI0，一路是SPI1。（ECU抽象层不同的ECU外围设备接口，可能底层都是使用同一种驱动）所以ECU抽象层就是对整个ECU产品的各种外围设备的封装。外围设备也是由MCU主芯片控制的，ECU抽象层是需要使用MCAL的接口的，作为抽象层，屏蔽了下层驱动的实现细节，将统一的接口，API暴露给上层从而实现功能，这样就做到了分层。上层调用falsh\_write接口的时候，会往flash中写一段数据，但是在这个过程中，它并不知falsh\_write到这个接口，再到这个MCU的时候，到mcal这一层具体调用的是什么驱动。

然后flash\_write在这一层直接是调用的SPI通信的驱动，会做封装。ECU抽象层从上层抽象MCAL层并提供访问内部和外部的服务。

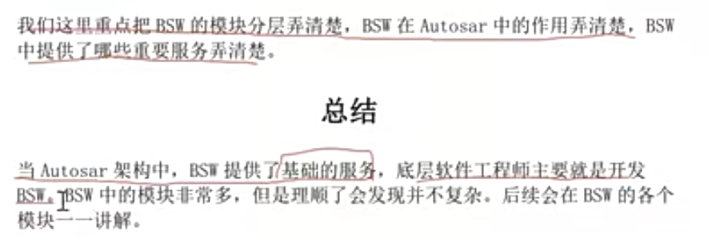
服务层将底层提供的服务封装起来供应用层使用，比如通信服务，存储服务，os操作系统服务等。就通信服务而言，应用层只需要调标准通信服务接口，比如CAN总线需要发一个信号，那么应用层只要调Com\_write接口，第一个参数写信号ID，第二个参数写信号值。这里不管信号是CAN、LIN、ETH，有信号ID和值就够了，因为服务层向上层提供了统一的接口。CDD复杂驱动有些模块不适用于Autosar的协议栈，在实际项目中会有很多模块，或者自己加一些外围的小芯片，但是这些并不好加到Autosar的标准架构中去。那就手写一些裸机的代码，然后封装成CDD的模块，然后再应用层直接调用函数的驱动，直接使能驱动就好了。比如说买了vector的工具链，vector的工具链只提供了通信、存储、IO这些标准的Autosar驱动，但是产品里面需要一些“偏门”的驱动，平常都不怎么使用的，那么就写一个CDD模块，应用层直接调用就好了，其实本质上就是写一个裸机程序。

上述就是BSW的三个分层和CDD模块的概述。









BSW层的服务提供了IO功能、通信和诊断功能，存储功能，看门狗功能，还有各种其它的驱动功能。BSW、ECUM模式管理属于服务层的内容。

看门狗：检测MCU，当MCU挂死的时候可以进行复位重启。芯片中的CPU和看门狗用的是2套时钟源。CPU在正常的运行过程中，看门狗是个计数器0-100。CPU在正常运行的过程中，当看门狗加到20的时候，会给看门狗复位。保证看门狗永远不会加到100，正常运行过程中，看门狗是永远不会加到100的。当CPU挂死了，不运行了，由于是2套时钟，看门狗会一直往外加，加到100之后，这时候看门狗会认为CPU已经挂失了，会把整个MCU芯片进行复位，然后将MCU芯片重启。

还有存储服务，就是提供数据写到EEPROM。

通信和诊断服务：提供CAN、LIN、ETH等通信和诊断功能

I/O功能：提供通用GPIO读写功能，ADC、PWM等特殊PORT外设功能。

OS服务：提供基础OS服务，任务周期运行、调度等功能

BswM：管理整个Bsw的模块

EcuM：管理ECU上下电等功能