搭载MetaOS操作系统的工智机在多个行业的项目案例

基于Xenomai

中科时代 宋健玮

目录

搭载MetaOS的工智机及行业应用案例

01

中科时代工智机介绍

新品介绍

带轴能力对比

Xenomai与preempt_RT对比

抖动对比

02

行业应用案例

3C电子行业

半导体行业

激光行业

风电行业

03

常见技术问题及解决方案

问题一 CodesysRTE 读写文件带来的超时隐患

问题二 try...catch导致的超时隐患

问题三 Linux驱动带来的模式切换: 移植到RTDM

中科时代2024工智机新品简介

2023年

EPC

SX21

Hygon C86 3330 4核 3.0GHz

SX5164

Intel x6425RE 4核 1.9GHz

2024年

IPC

SP7000Se

Intel N97 4核

SP7000H

Intel i5 12400
6核
2.5/4.4GHz(睿)

SP60

Intel E3845 4核 1.91GHz

SP7020H

SP5040A

Intel i9-12900K 8P8E 16核

SP5010A

Hygon-3350 4核 3.0GHz

EPC

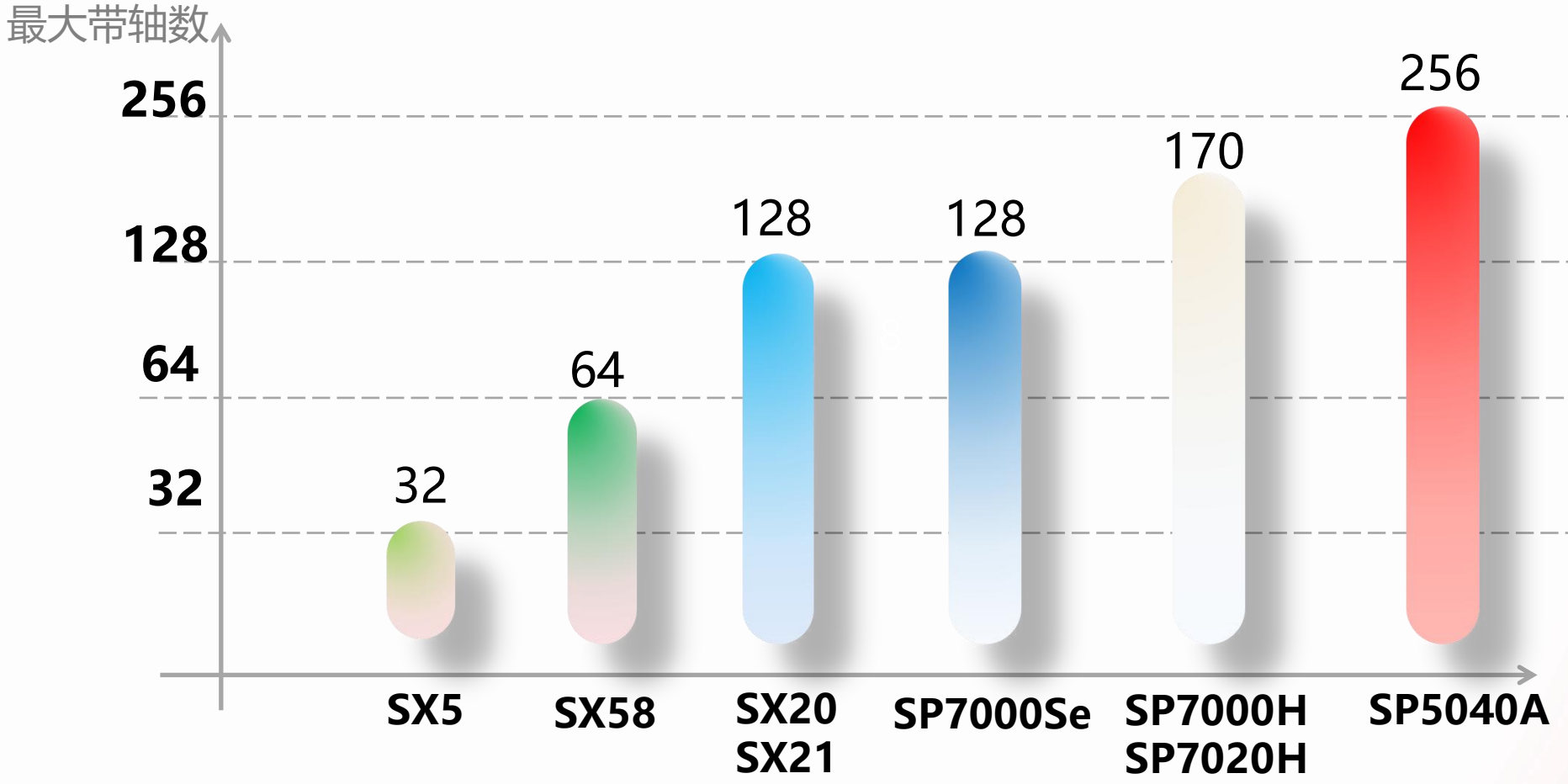
SX58

Intel Celeron
J6412

带轴能力对比

1msDC同步 抖动<30us

任务周期均为 **1ms** 的情况下



Xenomai 与 PreemptRT 延迟对比

SX5164

Intel x6425RE 4核 1.9GHz

测试环境

软件版本:

Linux Kernel: 5.15.51

Xenomai: v3.3

EtherCAT: IGH

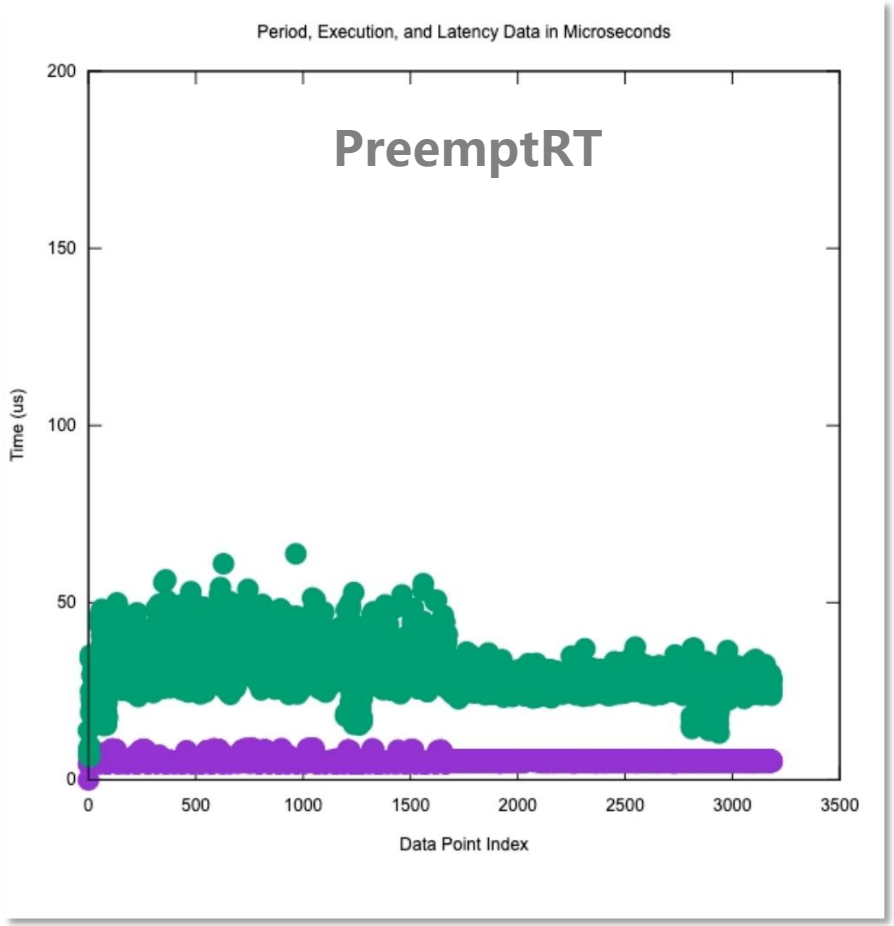
网卡驱动：rt_igc

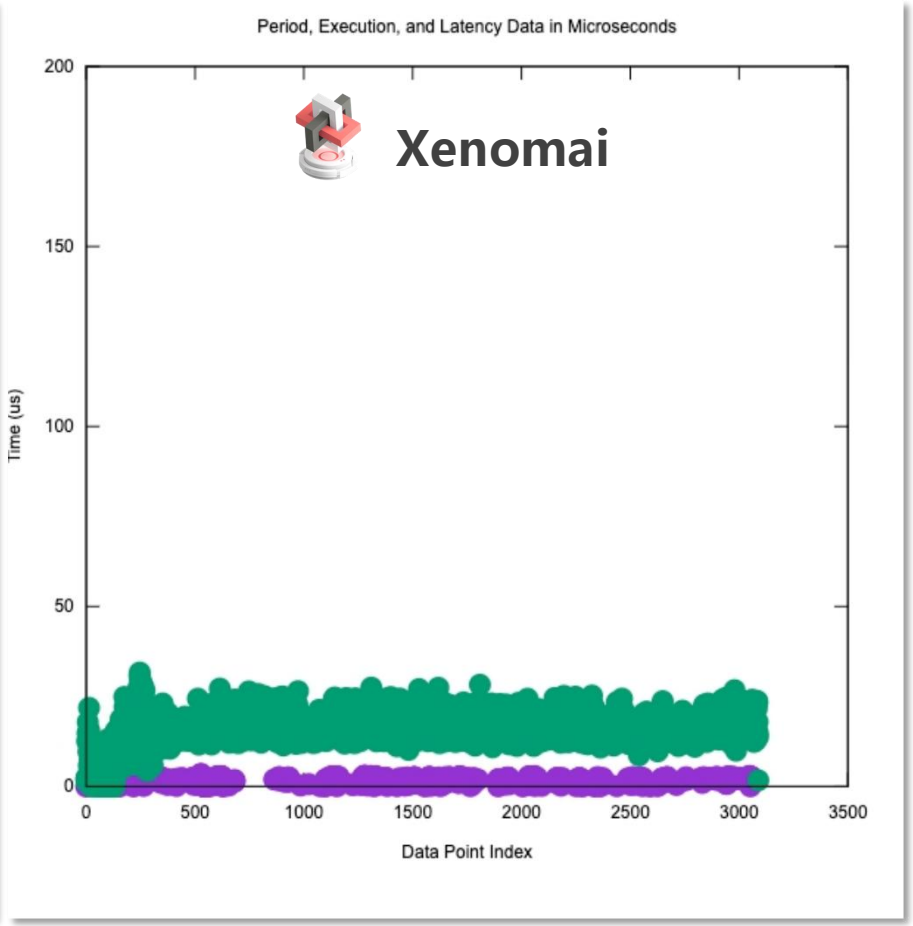
硬件平台:

中科时代SX51工智机

网卡芯片Intel i225-IT

结论：Xenomai 的延迟范围比PreemptRT更窄，抖动更低。

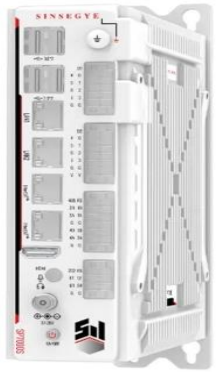




Latency Min(us)

Latency Max(us)

Xenomai在SP7000Se上72小时满负载稳定抖动测试



SP7000Se

Intel n97 4核 1.9GHz

测试环境

硬件平台

中科时代SP7000Se工智机

网卡芯片Intel i210

软件版本

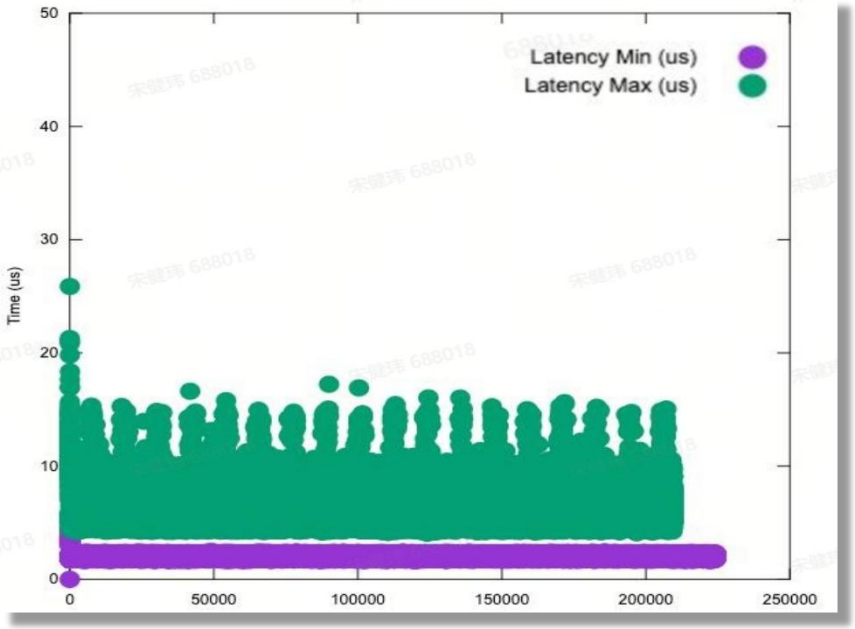
Linux Kernel: 5.15.51

Xenomai: v3.3

EtherCAT: IGH

网卡驱动: rt_igb

结论: SP7000Se 在CPU和GPU加满压力的情况下， 依然能长时间把延时控制在稳定范围内，抖动稳定在20us范围内。



SP7000Se 72小时满负载单轴运动jitter

Igh 使用xenomai API + RTDM 网卡驱动后，**完全避免掉非实时上下文切换**

```
Every 1.0s: cat /proc/xenomai/sched/stat

CPU  PID    MSW        CSW        XSC        PF    STAT       %CPU  NAME
  0  0      0          714036     0          0     00018000   99.5  [ROOT/0]
  1  0      0          276641260  0          0     00018000  100.0  [ROOT/1]
  2  0      0          5788321    0          0     00018000  100.0  [ROOT/2]
  3  0      0          0          0          0     00018000  100.0  [ROOT/3]
  0  3540270 60         61         65         0     000680c0    0.0  igh_ec_xenomai_
  0  3540273 1          31959      207788     0     00048044    0.5  igh_ec_xenomai_
```

3C行业：4轴Scara机械臂锁螺丝

需求与难点

客户诉求

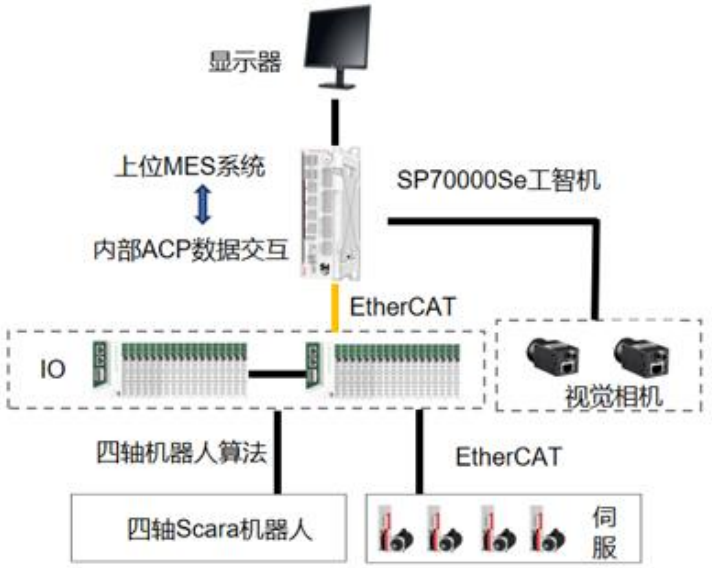
- 自动流水线上料
- 自动视觉相机拍照定位
- 4轴Scara机械臂稳定控制

业务形态

SCARA机器人通常具有4个轴。它们有两个平行臂，可以在一个平面内移动。最后一个轴与其他轴成直角。

解决方案

- 一台运行中科时代自研Meta OS操作系统的SP7000Se工智机替代原上位工控机+PLC，实现算控一体
- 实时域带轴以及带IO， 运行四轴机器人算法
- 上位MES系统通过ACP通讯与Meta OS非实时域进行高速数据交互，同时非实时域还带视觉检测



实施成效

搭载Xenomai的SP7000Se工智机带来的优势

1 生产效率效率提升

10秒/PCS 提升至：

8秒/PCS

2 视觉检测精度提升

提高：

产品质量

半导体行业：固晶机

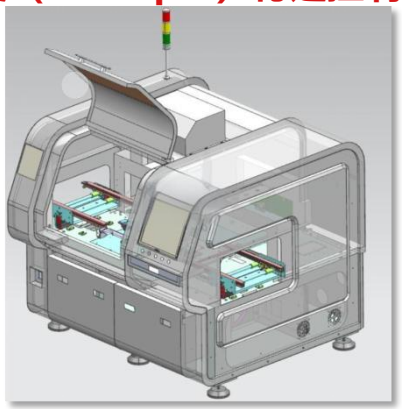
需求与难点

客户诉求

实现上料、点胶、送料、视觉缺陷检测、晶圆台、BONDING、上位MES通讯、日志存储等功能

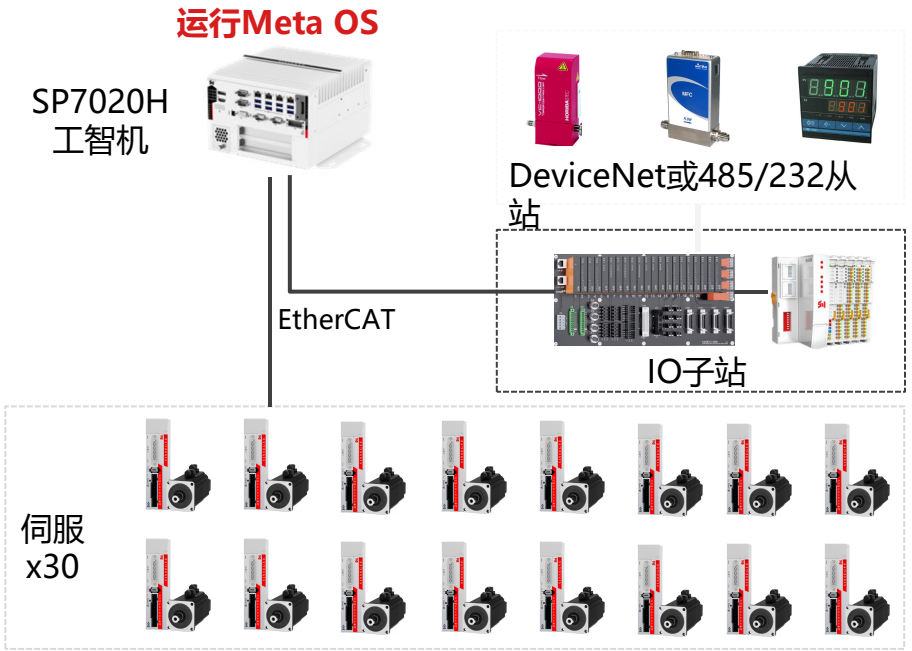
业务难点

实际产能 UPH>18K/hr
(units per hour)
Cycletime:500us
带多轴、带视觉并实现高速高精度（50-25μm）稳定控制



解决方案

- 一台运行中科时代自研Meta OS操作系统的SP7020H工智机替代原上位工控机+PLC，实现算控一体
- 实时域带轴以及带IO， 高速周期500us
- 非实时域视觉检测缺陷，开通 SECS-GEM 通讯协议与上位MES系统通讯，报警及参数修改日志记录存储3年以上



实施成效

依托Xenomai的可靠稳定高实时性架构，实现：

1 降低成本

算控一体 架构，节省成本

30%

2 指标突破

定位精度

25μm

3 易用开放

支持丰富的通信协议及内嵌功能库拓展，按需移植程序等

激光行业： 单晶硅激光切割取样机

需求与难点

客户诉求

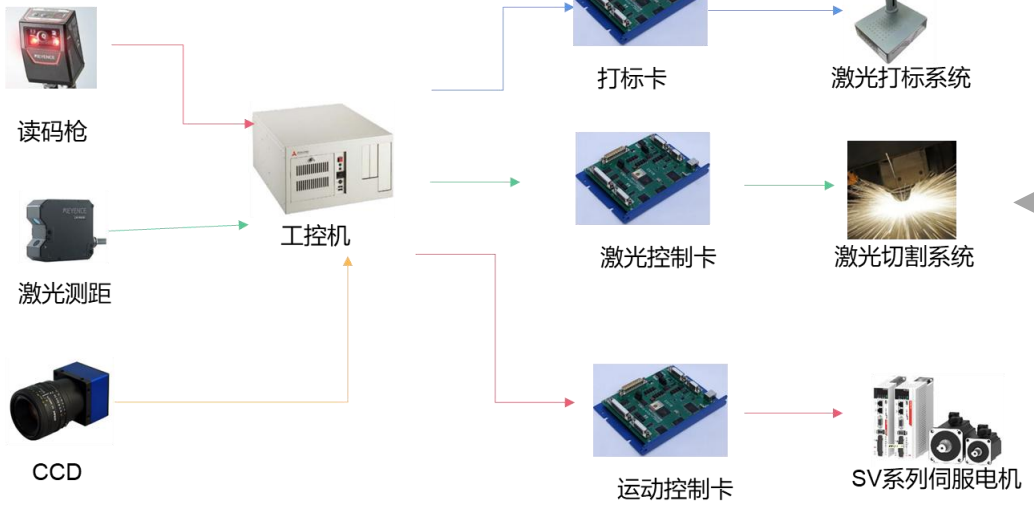
- 设备可涵盖220-340mm各种直径来料，厚度1-2.5mm。
- 来料会有各种破损，最多可到1/3缺失度，且整面厚度均匀性1mm。
- 切割取料25-50mm边长的正方形，上面的打标深度>0.1mm。
- 连线研磨机，联机可调。

业务难点

- **客户来料的多样性**
- **综合性激光设备工序复杂：要完成读码、CCD视觉识别、轴控、激光打标、激光切割。**

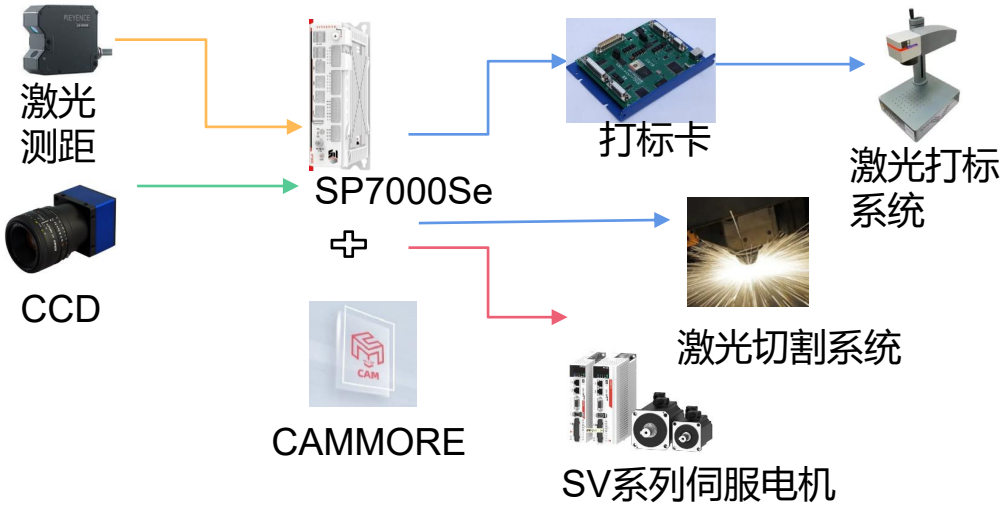
激光行业: 单晶硅激光切割取样机 - 新旧架构对比

传统上下位工控机+控制卡、运控卡



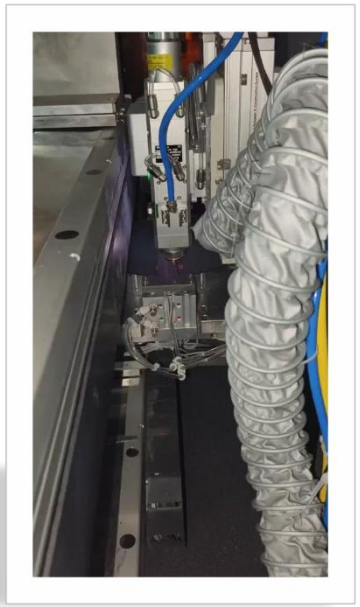
VS

搭载Xenomai的双内核架构



架构改进带来的优势:

- 一机多能，算控一体
- 成本降低 60%
- 双域隔离避免硬死机

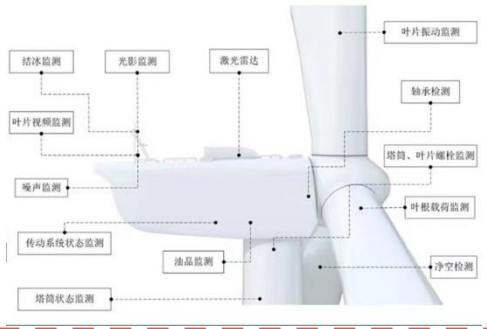


风电行业：风机主控

需求及难点

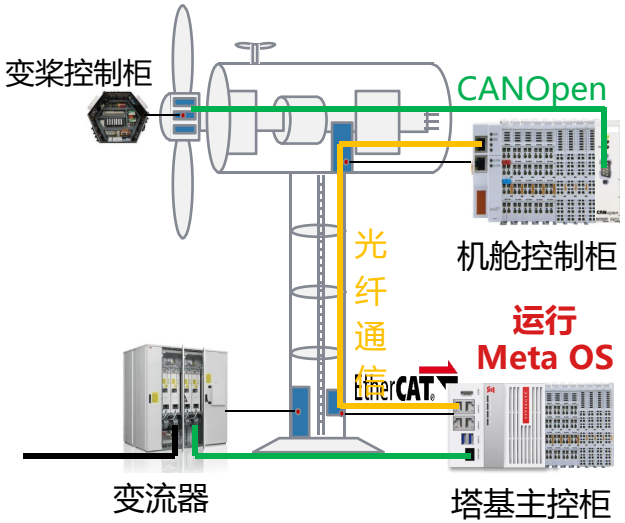
风电风机监控

- 叶片、塔筒、螺栓等高风险、高价值设备监测，包含传感器、采集、传输
- 通过数据，运行全状态CMS诊断平台进行故障判断、诊断、预测性维护等

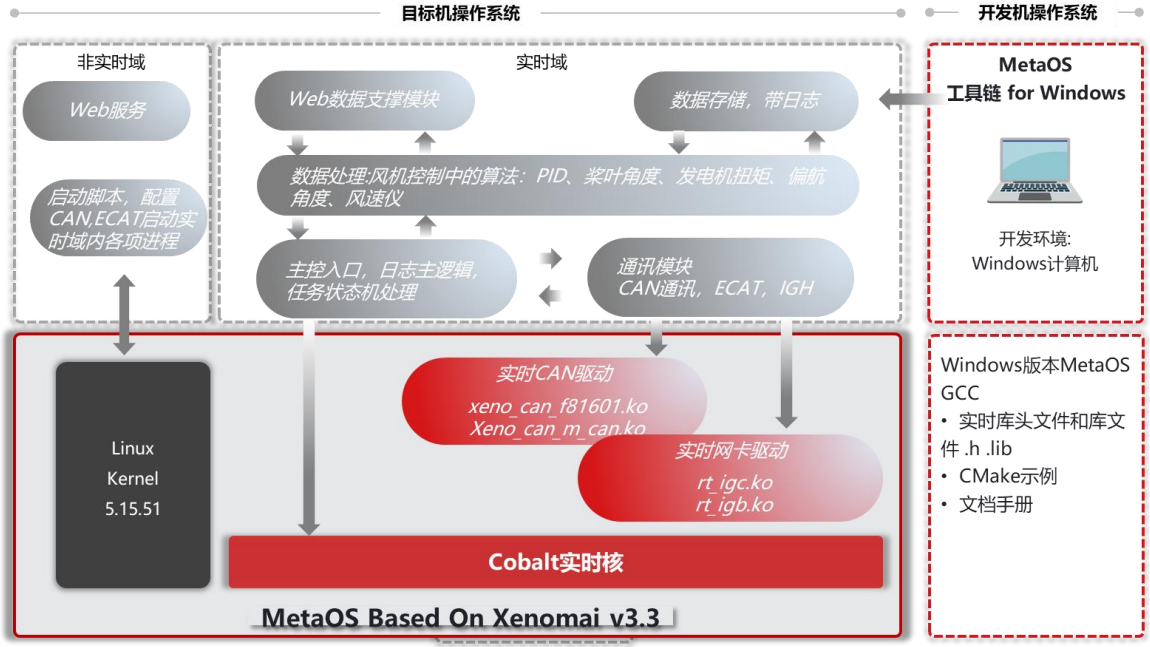


解决方案

- 提供VSCode编程环境下的工具链
- 支持多语言，多插件。编程接口支持C/C++，Matlab/Simlink等
- 实时核API手册，支持用户程序开发



易用且开放
无IDE自行移植和开发风电程序

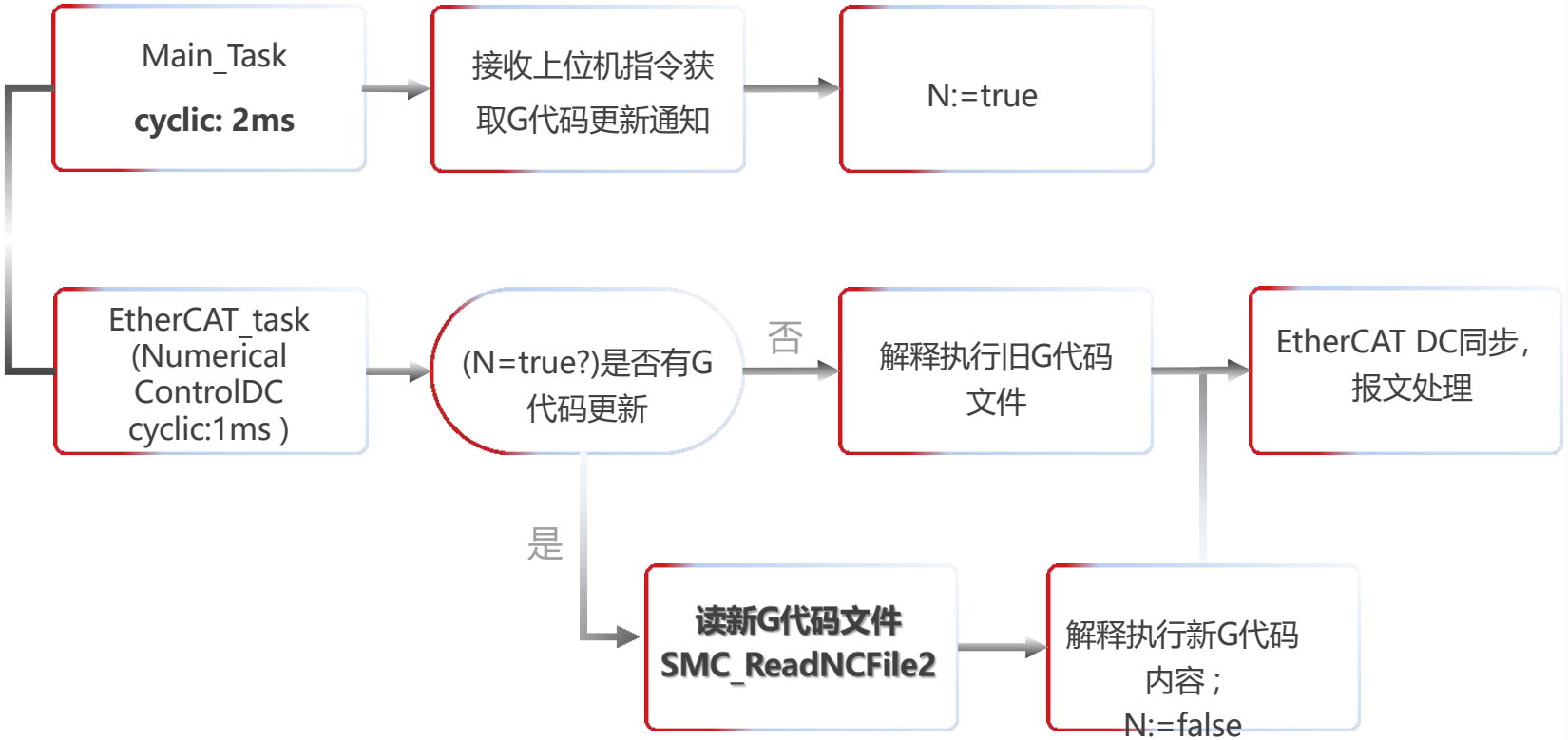


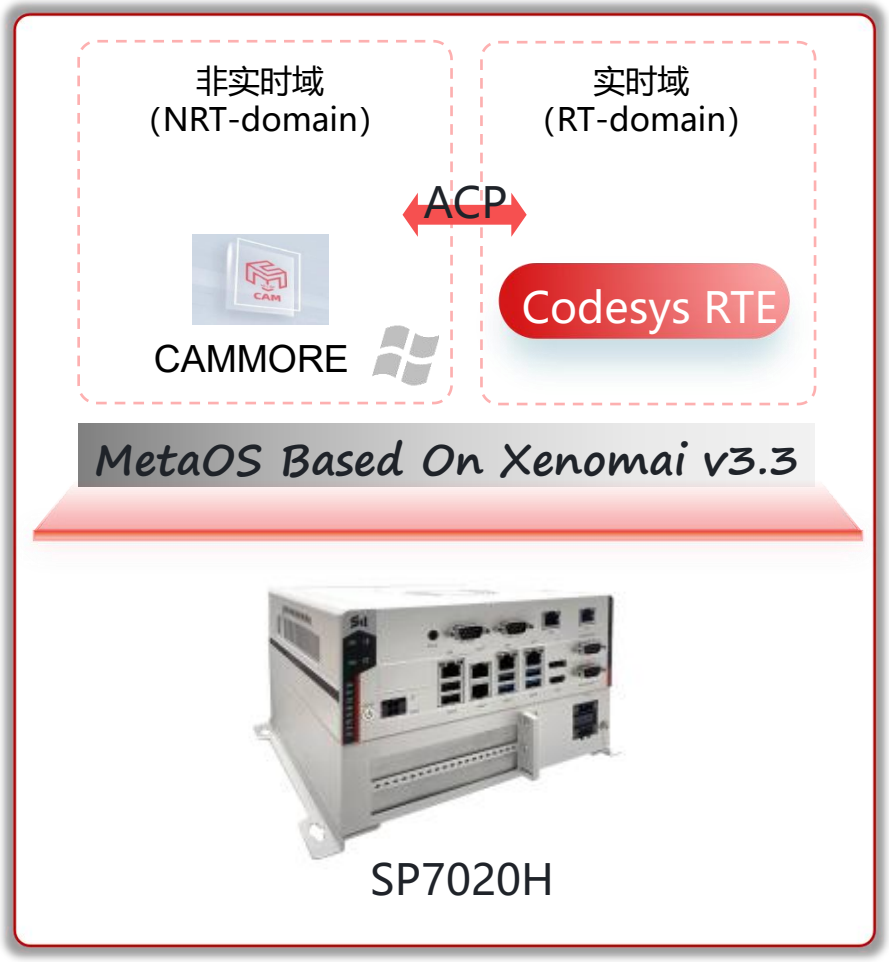
- 红色边框部分：MetaOS提供给客户的产品
- 灰色边框部分:用户使用MetaOS C/C++工具链编译出的应用

问题一: CodesysRTE 读写文件带来的超时隐患-现象

现象： 某项目Codesys RTE里有两个任务 EtherCAT_Task和Main_Task， EtherCAT_Task里，会判断是否有G代码文件更新，若有则读取新的G代码文件(功能块SMC_ReadNCFile2)，解释G代码并执行获得运控坐标点，通过EtherCAT通讯给伺服驱动器发指令，让机械臂运动到指定坐标。

但实际使用过程中，大约每3~5小时偶尔随机发生DC同步丢失，进而EtherCAT通讯中断。





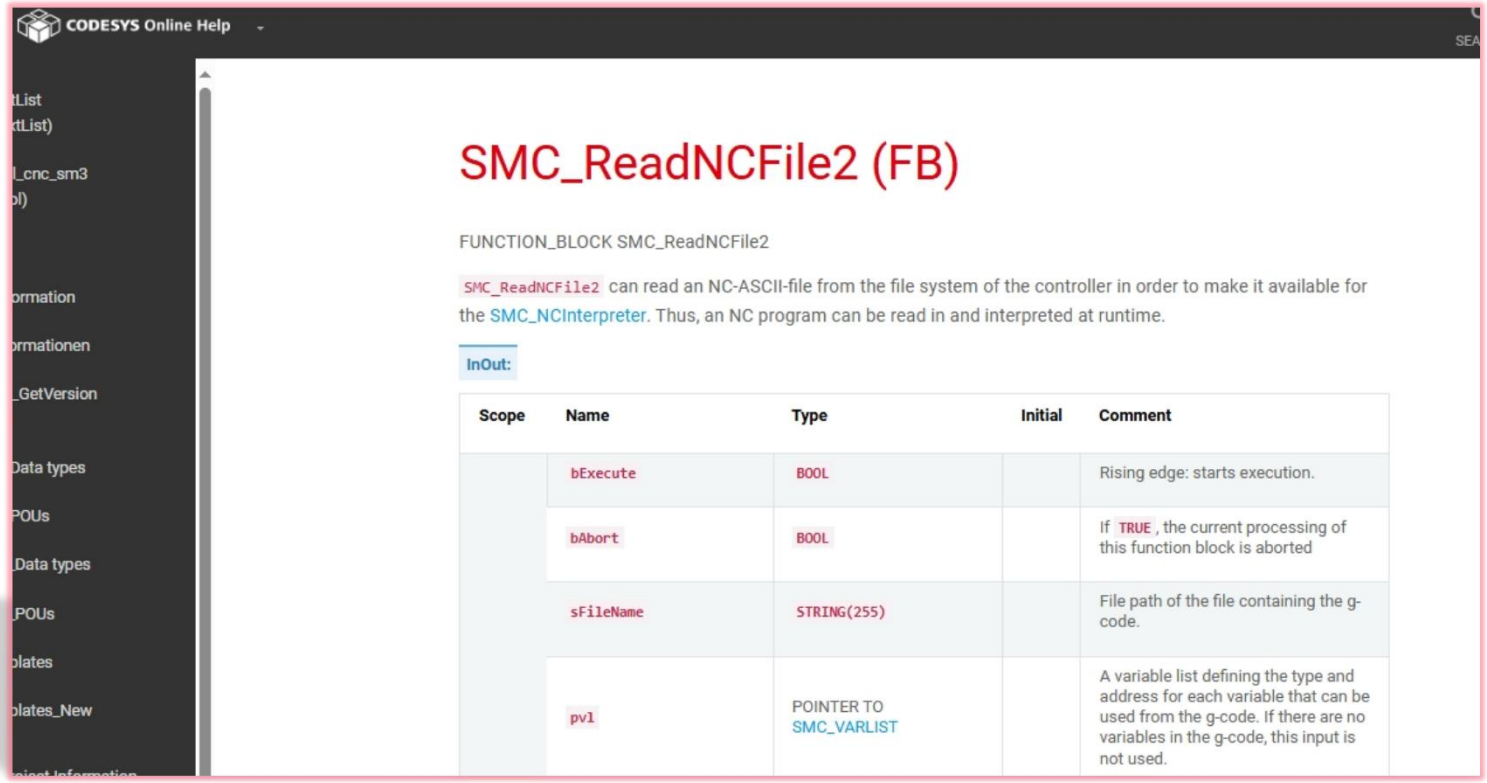
问题一: CodesysRTE 读写文件带来的超时隐患-释义

模式切换（Mode Switch）
模式切换是指线程从实时模式切换到非实时模式的过程。
这种切换通常会引入不可预测的延迟，严重影响系统的实时性能。

常见的三种模式切换情况

- **读取文件**
- 访问驱动程序
- 动态分配

SMC_ReadNCFile2：从NC-ASCII文件中读出G代码语句，并发送语句到G代码解释器



- **磁盘 I/O**：读取文件需要访问磁盘，而磁盘 I/O 操作的延迟不可预测，可能会阻塞线程。
- **系统调用**：调用诸如 open、read、lseek 等标准文件操作函数，这些函数属于 Linux 的系统调用，会导致线程从实时模式切换到非实时模式。

问题一: CodesysRTE 读写文件带来的超时隐患-改进

项目级改进方案

产品级改进方案

在中科时代MetaFacture中采用中科时代的文件操作库替换Codesys自带的异步文件操作库，保持用户原有代码和编程习惯的情况下，做到不切换模式。

MetaFacture

用"中科文件库"替换全部codesys自带的"文件功能库"

CAA File

- Enums
  - ATTRIB (Enum)
  - ERROR (Enum)
  - MODE (Enum)
- Function Blocks
  - Directory
    - DirClose (FunctionBlock)
    - DirCopy (FunctionBlock)
    - DirCreate (FunctionBlock)
    - DirList (FunctionBlock)
    - DirOpen (FunctionBlock)
    - DirRemove (FunctionBlock)
    - DirRename (FunctionBlock)
  - File
    - Close (FunctionBlock)
    - Copy (FunctionBlock)
    - Delete (FunctionBlock)
    - EOF (FunctionBlock)
    - Flush (FunctionBlock)
    - GetAttribute (FunctionBlock)
    - GetPos (FunctionBlock)
    - GetSize (FunctionBlock)
    - GetTime (FunctionBlock)
    - Open (FunctionBlock)
    - Read (FunctionBlock)
    - Rename (FunctionBlock)
    - SetPos (FunctionBlock)
    - Write (FunctionBlock)
- Functions
  - GetProperty (Function)
- Global Variables
  - CAA_Globale_Constants (GVL)
- Structs

问题二: try...catch导致的超时隐患-现象和解释

现象： 某项目采用Codesys Softmotion 功能块， 在EtherCAT任务中，

但实际使用过程中，大约每3~5小时偶尔随机发生DC同步丢失，进而EtherCAT通讯中断，日志能看到报错"Excetion Error"

小概率出现DC同步超时，日志报错Excetion Error ... 经查，代码内有try ... catch 语句。

解释：代码内有try ... catch 语句，造成 空指针异常被__TRY捕获， 最终在_CATCH里做了处理，导致实时任务延迟。

运算符：__TRY、__CATCH、__FINALLY、__ENDTRY

这些运算符是从 IEC 61131-3 标准扩展而来的，它们用于 IEC 代码中的特定异常处理。

句法

```
__TRY
    <statements_try>
__CATCH(exec)
    <statements_catch>
__FINALLY
    <statements_finally>
__ENDTRY
    <statements_next>
```

当声明中的 `__Try` 运算符抛出异常，应用程序不会停止。相反，它执行下面的语句 `__Catch` 因此开始异常处理。然后它执行下面的指令 `__FINALLY` 异常处理以 `__ENDTRY` 然后应用程序执行下一条语句。

异常的 IEC 变量具有数据类型 `__System.ExceptionCode`.

```
1  VAR
2      test            : bool
3      a, b            : ulint;
4  END_VAR
5
6  __TRY
7      IF test THEN
8          a:= 1/b;
9      END_IF
10     a:=2;
11 __CATCH
12     a:=111;
13 __ENDTRY
```

当b为0时，此代码发生除0异常

问题二: try...catch导致的超时隐患-改进

项目级改进方案

要求工程上，保持高实时任务里不允许 try ...catch 语句，来避免CPU进入异常处理。

产品级改进方案

在**MetaFacture**上，对try... catch语句做一些限定，用户编程时，若在高实时任务使用try...catch可给出报错提醒?

问题三: Linux驱动带来的模式切换: 移植到RTDM

中科时代 SINSEGYE

常见的三种模式切换情况

读取文件

访问驱动程序

动态分配

实时CAN驱动
xeno_can_f81601.ko
xeno_can_m_can.ko ✓

实时网卡驱动
rt_igc.ko
rt_igb.ko ✓

项目常见需求优先级高已支持

实时GPIO ⏳

实时串口
(RS485, RS232) ⏳

实际需求优先级不高待支持

访问驱动程序

问题描述

在实时线程中，调用一些标准的系统调用，如 open、read、write、ioctl、socket、connect、sendto、recvfrom 等，可能会导致模式切换。这是因为这些系统调用通常涉及到与 Linux 内核的交互，**可能会阻塞或引入延迟，迫使实时线程切换到非实时模式**，从而影响系统的实时性。

原因分析

阻塞操作：标准的驱动程序可能在 I/O 操作中发生阻塞，等待硬件响应或资源可用。
非确定性：这些系统调用的执行时间可能不可预测，取决于系统状态、硬件性能等因素。
内核空间交互：涉及到与 Linux 内核的交互，而非实时的内核服务可能无法满足实时性的要求。

解决方案

为了解决上述问题，可以采用以下方法：

使用基于 Cobalt 的驱动程序框架 RTDM

RTDM（Real-Time Driver Model） 是 Cobalt 提供的实时驱动程序模型，旨在简化实时驱动程序的开发。

感 谢 聆 听