



Using the Common Device Interface in TINE

Philip Duval and Honggong Wu

this.Contens()

- A few words about driver support
- Motivation behind CDI; Design goals
- How CDI works
- CDI manifest; CDI database
- A few server examples using CDI
- Status

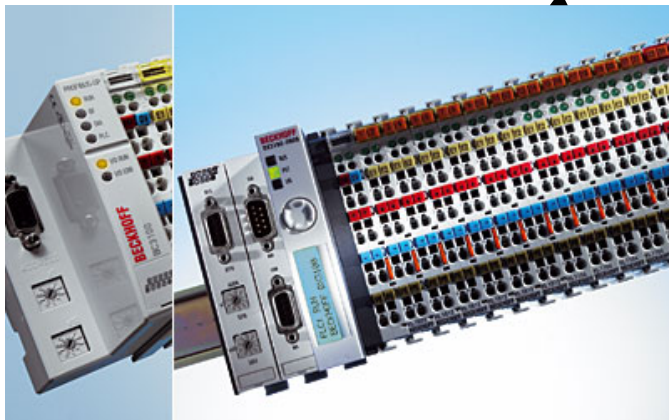
hardware support (1)

- do it yourself
 - use the drivers that come with what you bought, or write your own
 - mostly sedac for HERA + pre-accelerators (+ gpib, rs232, increasing use of can, dsps ...)
 - TINE via LabView (leverage the LV drivers)
 - different interface API: bus and operating system dependent

hardware support (2)

○ PETRA III

- mixed
- more emphasis on CAN, TwinCat, + SEDAC, GPIB, rs232, vme, siemensPLCs, Libera etc.



<http://www.beckhoff.com>



<http://www.i-tech.si/>



Application Programmers

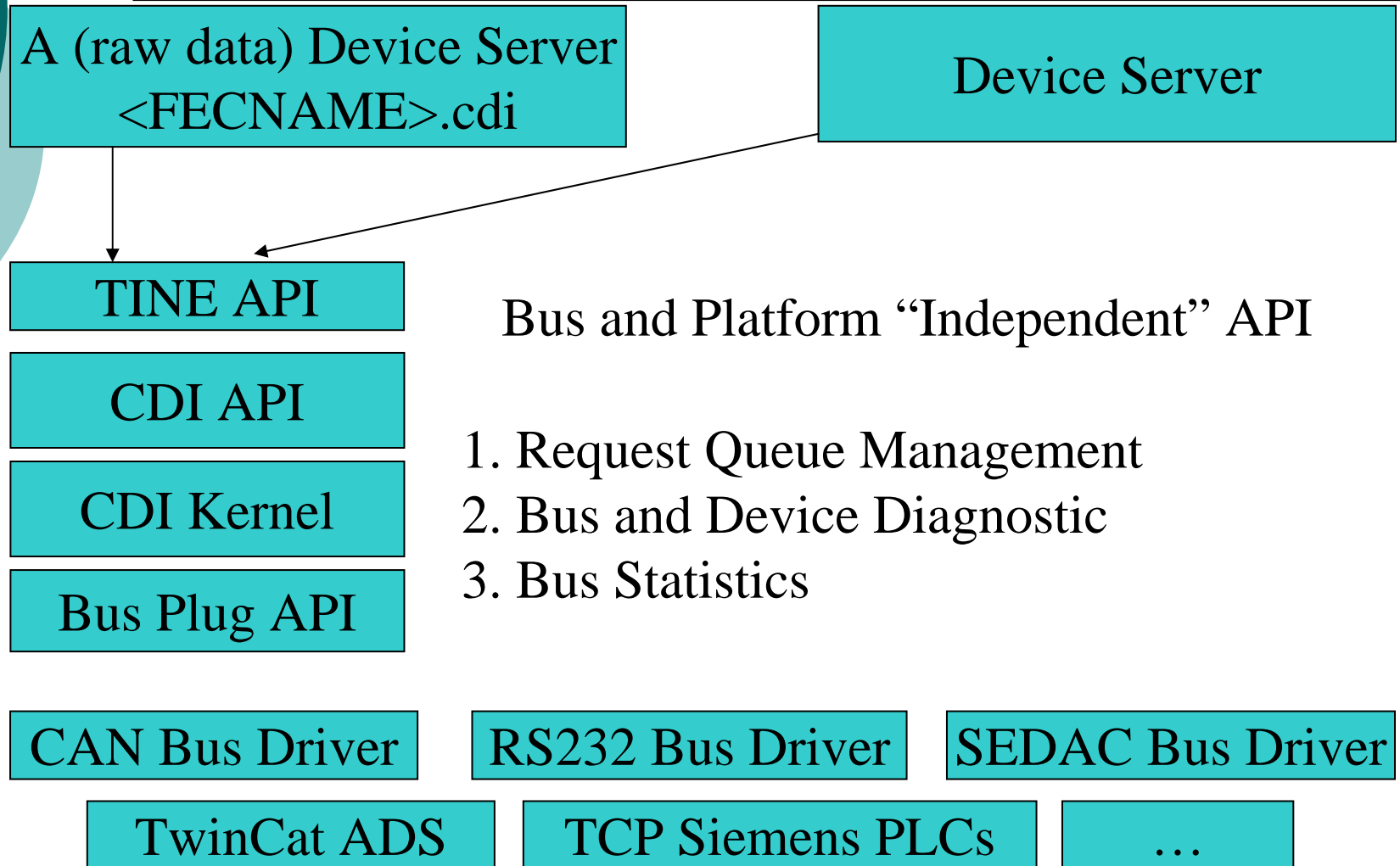
- either learn an interface API for each hardware card of a bus, and each operating system
- or create a common interface for device access

- use TINE CDI !

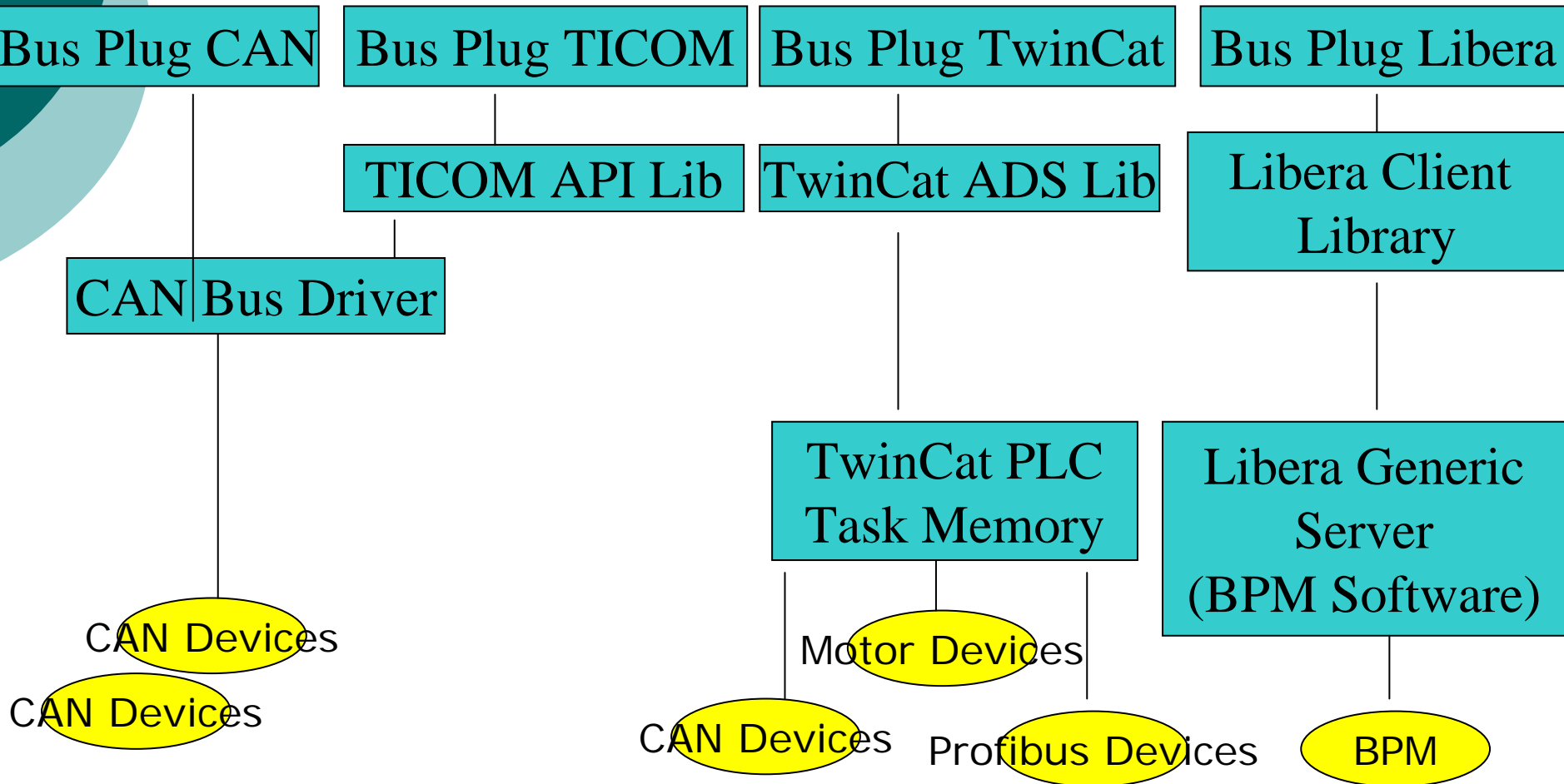
CDI API

- have a general and powerful client API in TINE
 - use it for accessing the hardware => no new API to learn!
- Allow:
 - the good old "do it yourself" method if you want to or need to
 - direct CDI API interface (TINE similar for device registration and device access)
 - direct TINE client calls for accessing hardware devices + database registration of devices.

Device Server using TINE CDI



CDI Bus Plug?



CDI Bus Plug continue...

- For a new Bus type, a few functions need to be implemented, use the driver API for the target platform
 - Open Bus, initialization
 - Close Bus, cleanup
 - Read/Write Data
 - Filter for input parameters
 - Bus Scan
 -

Synchronous Calls

/<context>/<server>/<device>
e.g.: "/PETRA/Vacuum/WLB.HP141"
or : "/localhost/cdi/pump1"

Device Property or Method
e.g.: "Pressure"
or: "RECV.CLBR"

ExecLink(devName, devProperty, dout, din, access, timeout)

Output Data object
returned from
Server

Input Data
Object sent to
Server

Access flags:
READ, WRITE, +
misc.

timeout in msec

Atomic !

Asynchronous Calls

Analogous to synchronous parameters ...

AttachLink(devName, devProperty, dout, din, access, pollrate,
void (*callback)(int,int), callbackID, mode)

Callback with callback id
and status code ...

CM_CANCEL
CM_SINGLE
CM_REFRESH
CM_POLL
CM_NETWORK
CM_GROUPED
CM_WAIT
+ ...

Atomic !

CDI API Details

- **Device Name** (name or number):
 - `"/localhost/cdi/#1"`
 - `"/localhost/cdi/#1-#100"`
 - `"/localhost/cdi/#1,#3-#10,#99"`
 - `"/localhost/cdi/pump1 – pump100"`
 - ...
- **Device Properties** (methods)
 - `"RECV"`
 - `"SEND"`
 - `"RECV.SEND.ATOM"`
 - `"SEND.RECV.ATOM"`
 - `"RECV.CLBR"`
 - `"SEND.RECV.CLBR"`
 - `"ADDR"`
 - `"BUSNAME"`
 - `"BUSSCAN"`
 - `"BUSERRORS"`
 - ...

Can Use Device Name
or Device Number !

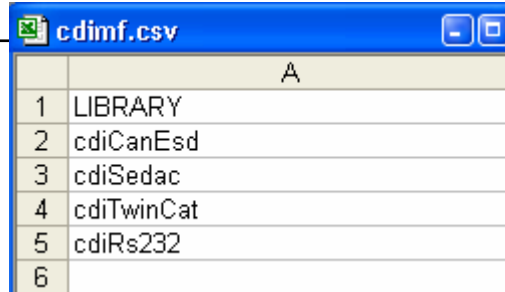
Read/Write Raw or
Calibrated data

Atomic pair-wise
access

Device information

CDI: How it works ...

1) Bus Manifest :



	A
1	LIBRARY
2	cdiCanEsd
3	cdiSedac
4	cdiTwinCat
5	cdiRs232
6	

← Bus Interface Plugs

2.) **cdiLoadLib**("cdiCanEsd.dll") - Windows
cdiLoadLib("libcdiCanEsd.so") - Unix
cdiLoadLib("cdiCanEsdLib.o") - VxWorks
Etc. ...

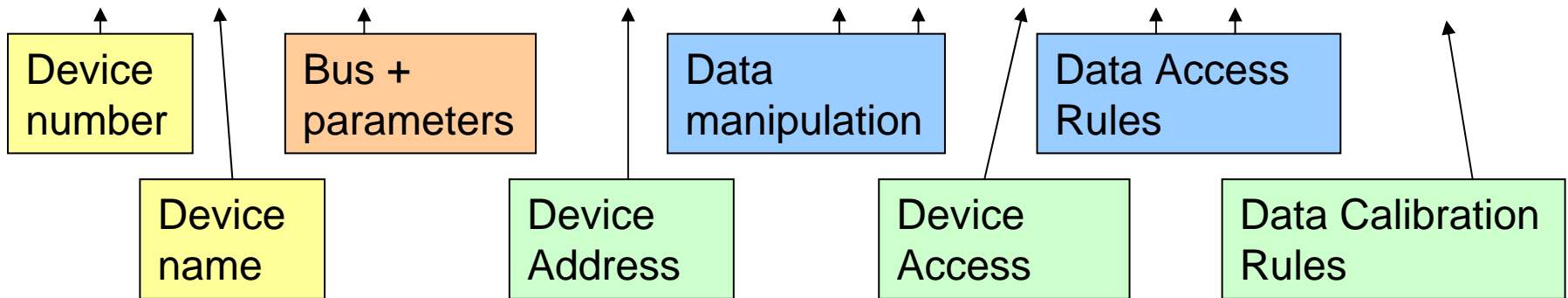
3). Library's prologue code 'plugs' into CDI:

```
int cdiRegisterBus(char *busName);  
int cdiRegisterBusInitialization(char *busName,int (*fcn)(int,int,int,char *));  
int cdiRegisterBusHandler(char *busName,void (*fcn)(CdiRequestInfoBlk *));  
int cdiRegisterBusCleanup(char *busName,int (*fcn)(int));  
int cdiRegisterBusFilter(char *busName,int (*fcn)(int *, int));  
int cdiRegisterBusScanner(char *busName,int (*fcn)(char *,char *, int));  
...
```

CDI: How it works ...

- Either register CDI devices dynamically via API or
- Use a CDI database to register devices at initialization:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	NUMBER	NAME	BUS	LINE	INDEX	ADDRESS	MASK	PATTERN	TOLERANCE	ACCESS	INPUT	FORMAT	LIMIT	RULE
2	1	SPS-WQPS	CAN:500	2	0	1:100:0:0:1:0	0xffff	0	0	WRRD	32	short	01:01	
3	2	SPS-WQPS1	CAN:100	1	0	1:100:0x600:0x3011:0:0	0xffff	0	0	READ		long		
4	3	SerialDevice1	RS232:9600:0:1:1:0:0:0	2	1		0xffff	0	0	READ		text		
5	4	Temperature1	SEDAC	1	0	16.252:0	0xffff	0	0	WRRD	8	short	01:01	"+120:*0.01;+100.0:*52:~.5"
6	5	Temperature2	SEDAC	1	1	16.253:0	0xffff	0	0	WRRD	12	short	01:01	"+120:*0.01;+100.0:*52:~.5"



CDI Initialization Flow ...

- Application calls `cdiInitialize()`
 - cdi library loads and reads manifest
 - each entry in manifest references a bus plug library
 - cdi calls `cdiLoadLibrary()` for each bus plug in the manifest
 - each bus plug loads()
 - bus plugs register all bus handlers with CDI
 - cdi looks for local database
 - Reads database
 - Registers individual device information
- Application can then make calls to CDI
 - via TINE client interface (`ExecLink()`)
 - via CDI interface (TINE similar) (`cdiExecLink()`)

CDI Examples

```
dout.dArrayLength = 100;
dout.dFormat = CF_INTINT;
dout.data.vptr = rbPressData; // <- INTINT[100] object
AttachLink("/localhost/cdi/#1-#100", "RECV.CLBR", &dout,
  NULL, 1000, cbPressData)
```

Reads devices 1 to 100, calibrates the raw data, fills in rbPressData[] and calls the callback cbPressData() at 1 Hz. (value + status pairs)

```
dout.dArrayLength = 100;
dout.dFormat = CF_UINT16;
dout.data.sptr = rbPressData;
AttachLink("/localhost/cdi/#1", "RECV.CLBR", &dout, NULL, 1000,
cbPressData)
```

Reads device 1 100 times, calibrates the raw data, fills in rbPressData[] and calls the callback cbPressData() at 1 Hz.

CDI Examples

```
din.dArrayLength = 1;  
din.dFormat = CF_UINT16;  
din.data.sptr = &setValue;  
ExecLink ("/localhost/cdi/#16", "SEND", NULL, &din, 1000)
```

Sends 'setValue' to device #16

```
din.dArrayLength = 1;  
din.dFormat = CF_UINT16;  
din.data.sptr = &setValue;  
dout.dArrayLength = 1;  
dout.dFormat = CF_UINT16;  
dout.data.sptr = &rbValue;  
ExecLink ("/localhost/cdi/#1", "SEND.RECV.ATOM", &dout, &din, 1000)
```

Sends 'setValue' to device #1 and reads rbValue from device #1 atomically.

BusScan Implementation Examples

- o Sedac, TwinCat and CAN

The image displays four instances of the 'Instant Client' software, each configured for a different BusScan implementation. Each window shows a configuration panel at the top and a terminal window at the bottom.

Instant Client (Instance 1): Configured for SEDUSB. The terminal shows the command string: `getcommands, getlines, getcrates, getmodules, readmodules, writemodules@`.

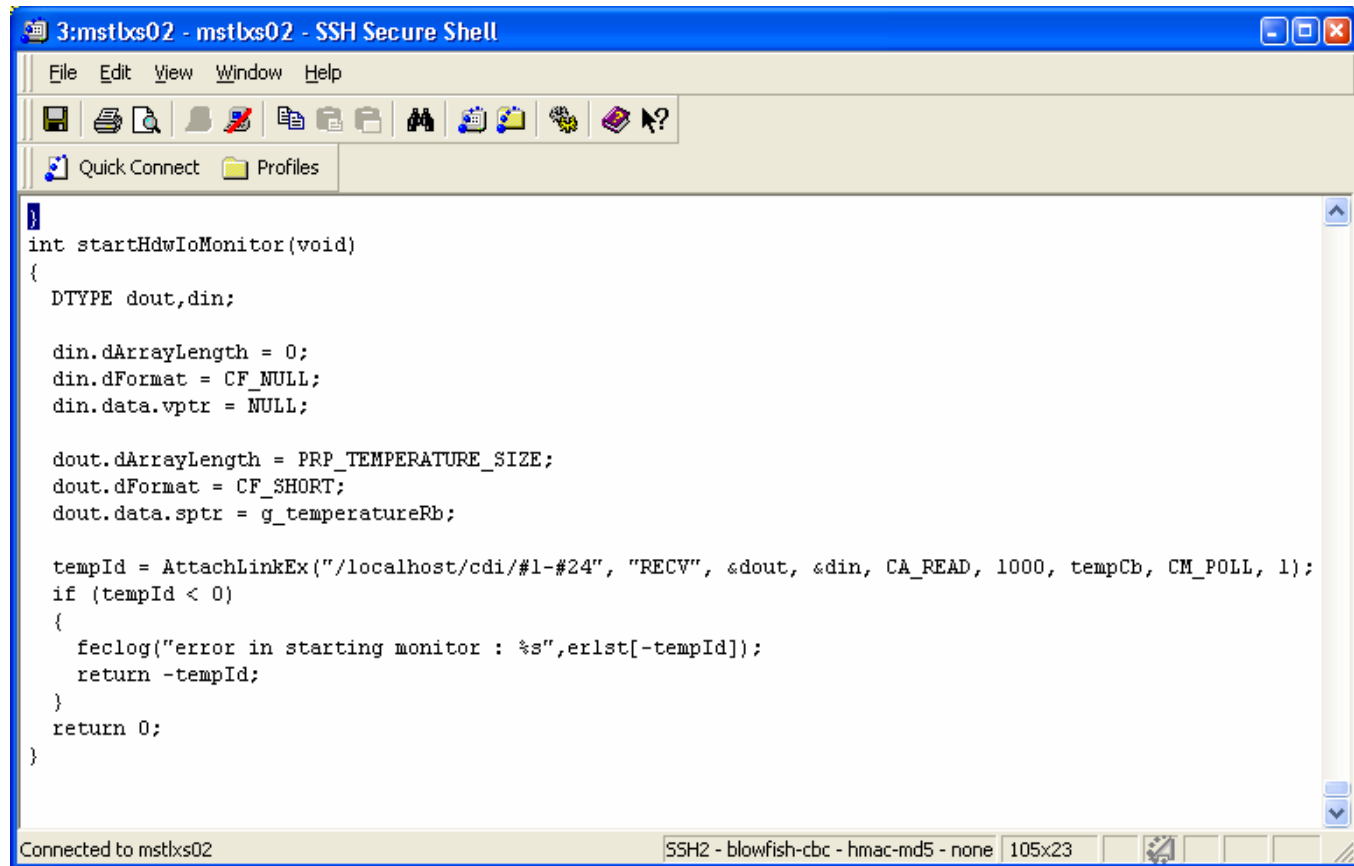
Instant Client (Instance 2): Configured for CANESD. The terminal shows the command string: `getcommands, getlines, getrcvmsg, getmodules, readsdo, writesdo, readpdo1, writepdo1, readpdo2, writepdo2, writenmt, readnmt, startremote, stopremote, enterpreoperation, resetnode, resetcommunicationp?`.

Instant Client (Instance 3): Configured for SIEMENS SPS. The terminal shows an error message: `Error < 38 > : RMT: not implemented yet`.

Each window also displays the following configuration details:

- Device Context: HARDWARE
- Device Subsystem: ALL
- Device Server: MSTXPWU1.CDI
- Device Name: SEDUSB, TWINCAT, CANESD, or SIEMENS SPS
- Device Property: BUSSCAN
- Data Size: 1024
- Data Type: TEXT
- Description: Scan bus according to input given
- Timeout: 1000

CDI Practical Example (C)



The image shows a screenshot of an SSH Secure Shell terminal window. The window title is "3:mstlx02 - mstlx02 - SSH Secure Shell". The menu bar includes "File", "Edit", "View", "Window", and "Help". The toolbar contains various icons for file operations and system functions. Below the toolbar, there are tabs for "Quick Connect" and "Profiles". The main area of the window displays C code for a function named `startHdwIoMonitor`. The code defines two data structures, `din` and `dout`, and uses `AttachLinkEx` to establish a connection to a local device. It includes error handling with `feclog` and returns `-tempId` on failure or `0` on success.

```
int startHdwIoMonitor(void)
{
    DTYPE dout,din;

    din.dArrayLength = 0;
    din.dFormat = CF_NULL;
    din.data.vptr = NULL;

    dout.dArrayLength = PRP_TEMPERATURE_SIZE;
    dout.dFormat = CF_SHORT;
    dout.data.sptr = g_temperatureRb;

    tempId = AttachLinkEx("/localhost/cdi/#1-#24", "RECV", &dout, &din, CA_READ, 1000, tempCb, CM_POLL, 1);
    if (tempId < 0)
    {
        feclog("error in starting monitor : %s",erlst[-tempId]);
        return -tempId;
    }
    return 0;
}
```

At the bottom of the window, the status bar shows "Connected to mstlx02" on the left and "SSH2 - blowfish-cbc - hmac-md5 - none 105x23" on the right.

CDI: Practical Example (Java)

```
public static void start()
{ // call this test method as an alternative to a background task
  // start up CDI ...
  Cdi.start();

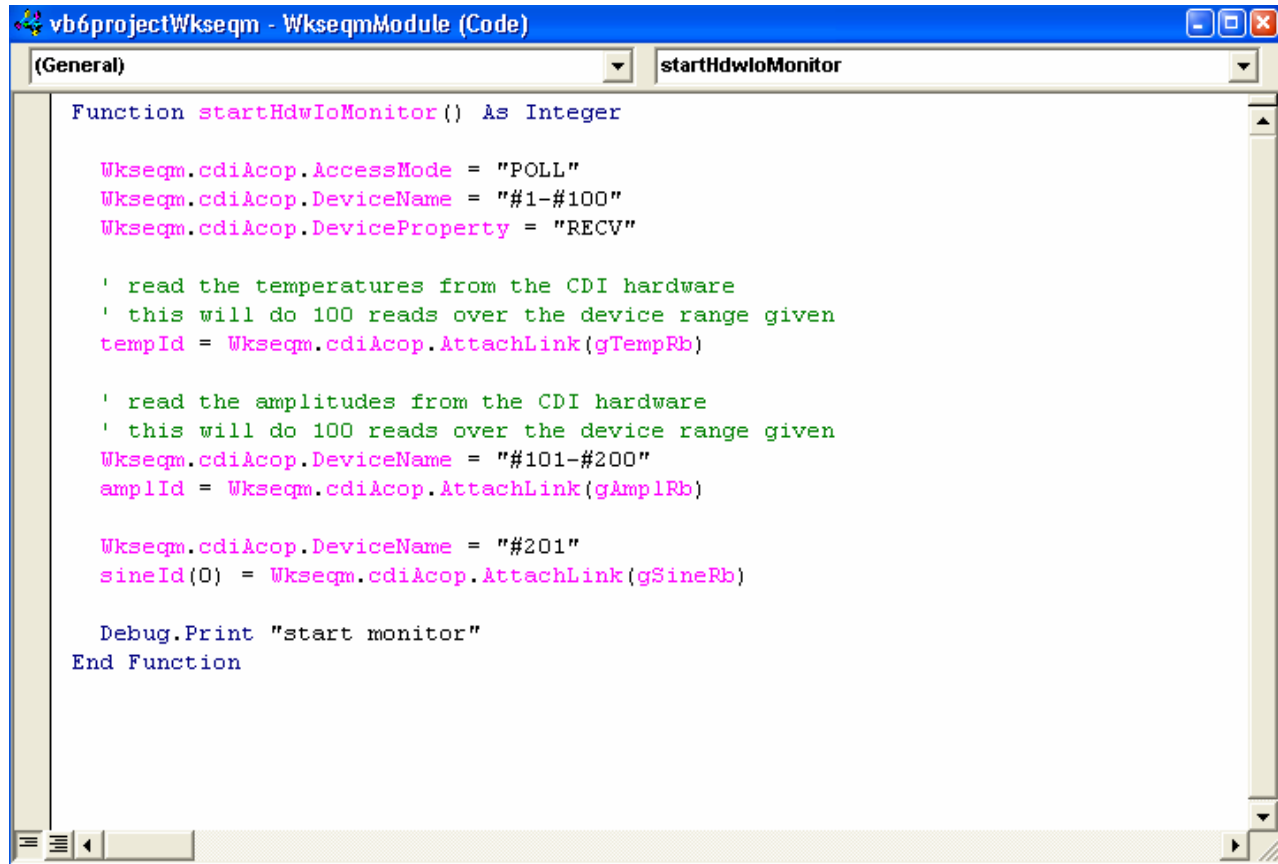
  // write something to some registered devices ...
  // using device numbers in the server code is probably preferable
  // as the maybe only the database knows what names were assigned ...
  setValue("#1",66);

  // start a read monitor on the temperature devices ...
  tempLink = new TLink(cdiPrefix + "#1-#100", "RECV", new TDataType(tempRb), null, TAccess.CA);
  HdwIoCallback tempCb = new HdwIoCallback();
  tempLink.attach(TMode.CM_POLL, tempCb, 1000);
  System.out.println("temp link status : " + tempLink.getLinkStatus());

  // start a read monitor on the amplitude devices ...
  amplLink = new TLink(cdiPrefix + "#101-#200", "RECV", new TDataType(amplRb), null, TAccess.CA);
  HdwIoCallback amplCb = new HdwIoCallback();
  amplLink.attach(TMode.CM_POLL, amplCb, 1000);
  System.out.println("ampl link status : " + amplLink.getLinkStatus());

  // start a read monitor on the sine devices ...
  sineLink = new TLink(cdiPrefix + "#201", "RECV", new TDataType(sineRb), null, TAccess.CA_RE);
  HdwIoCallback sineCb = new HdwIoCallback();
  sineLink.attach(TMode.CM_POLL, sineCb, 1000);
  System.out.println("sine link status : " + sineLink.getLinkStatus());
}
```

CDI: Practical Example (VB)



```
vb6projectWkseqm - WkseqmModule (Code)
(General) startHdwIoMonitor
Function startHdwIoMonitor() As Integer
    Wkseqm.cdiAcop.AccessMode = "POLL"
    Wkseqm.cdiAcop.DeviceName = "#1-#100"
    Wkseqm.cdiAcop.DeviceProperty = "RECV"

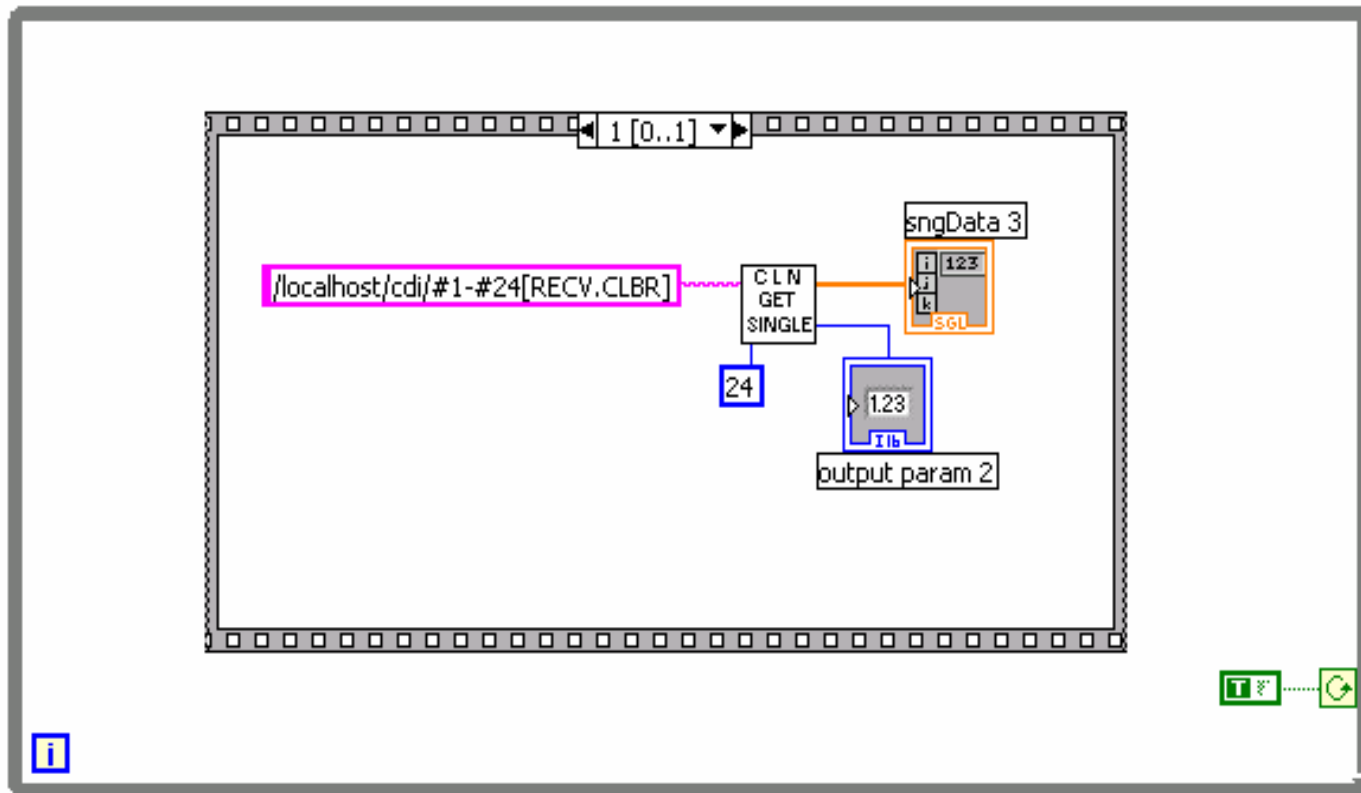
    ' read the temperatures from the CDI hardware
    ' this will do 100 reads over the device range given
    tempId = Wkseqm.cdiAcop.AttachLink(gTempRb)

    ' read the amplitudes from the CDI hardware
    ' this will do 100 reads over the device range given
    Wkseqm.cdiAcop.DeviceName = "#101-#200"
    amplId = Wkseqm.cdiAcop.AttachLink(gAmplRb)

    Wkseqm.cdiAcop.DeviceName = "#201"
    sineId(0) = Wkseqm.cdiAcop.AttachLink(gSineRb)

    Debug.Print "start monitor"
End Function
```

CDI: Practical Example (LabView)



CDI Remote

CDI automatically exports a (raw data) device server with the name <FECNAME>.CDI

The screenshot shows the 'Instant Client (Instance 2)' window. The interface includes a menu bar with 'Print ...', 'Options ...', 'Debug Tools ...', and 'Input Panel !'. Below the menu bar are several configuration panels:

- Device Context:** TEST
- Device Subsystem:** ALL
- Device Server:** PT100.1.CDI
- Device Name:** HE_TmpGrp1-8
- Device Property:** RECV
- Data Size:** 8
- Data Type:** INTEGER
- Description:** Issue CDI Read Telegram

A list of device properties is visible on the right side of the configuration area:

- BUSERRORS
- BUSNAME
- BUSNAMES
- BUSSCAN
- RECV (highlighted)
- RECV.CLBR
- RECV.SEND.ATOM
- SEND

The main data display area shows the following path and timestamp: /TEST/PT100.1.CDI/HE_TmpGrp1-8 RECV @ Jan 26 11:25:20.670. Below this, a list of data points is shown:

(0)	225
(1)	224
(2)	217
(3)	218
(4)	84
(5)	666
(6)	0
(7)	0

Additional controls on the right include 'Draw Mode' set to 'Text dun', 'Autoscale' (unchecked), and 'Log Scale' (checked).

CDI:Petra3/EMBL Motor Server with Beckhoff PLCs

- Bus Plugs (ADS DLL) for Beckhoff PLCs for Windows
- A PLC program (Motor and div IOs) in TwinCat

Use template in CDI database to register devices at initialization:

cdiaddr-III.csv												
A	B	C	D	E	F	G	H	I	J	K	L	
NUMBER	NAME	BUS	LINE	MASK	ADDRESS	ACCESS	INPUT	FORMAT	LONG_NAME	LIMIT	PATTERN	
11	0 MOTOR:Start	TEMPLATE	0					short	Motor[.Run	1:01		
12	0 MOTOR:CMD	TEMPLATE	0					long	Motor[.inCmd	6:04	!=1:04	
13	0 MOTOR:fitCMD	TEMPLATE	0					float	Motor[.inRealCmd	6:04	!=1:04	
14	0 MOTOR:TgtPos	TEMPLATE	0			RD		long	Motor[. SollPositionAbs	1:01		
15	0 MOTOR:CurPos	TEMPLATE	0			RD		long	Motor[.IstPositionAbs	1		
16	0 MOTOR:Type	TEMPLATE	0			RD		Short	Motor[.regAry[8]	1		
17	0 MOTOR:sync	TEMPLATE	0			RD WR		Short	Motor[.syncRunFlag	1		
18	0 MOTOR:Status	TEMPLATE	0			RD		Short	Motor[. Status	1		
19	0 MOTOR:mSteps	TEMPLATE	0			RD		Short	Motor[.microSteps	1		
20	0 MOTOR:rSteps	TEMPLATE	0			RD		Short	Motor[.fullSteps	1		
21	0 MOTOR:Rps	TEMPLATE	0			RD		float	Motor[.Rps	1		
22	0 MOTOR:Rms	TEMPLATE	0			RD		float	Motor[.Rms	1		
23	0 MOTOR:Rfs	TEMPLATE	0			RD		float	Motor[.Rfs	1		
24	0 MOTOR:mVeloc	TEMPLATE	0			RD		float	Motor[.travelVelocity	1		
25	0 MOTOR:rCurPos	TEMPLATE	0			RD		float	Motor[.rlstPosition	1		
26	0 MOTOR:rTgtPos	TEMPLATE	0			RD		float	Motor[.rSollPosition	1		
27	0 MOTOR:maxVel	TEMPLATE	0			RD		float	Motor[.Max_Velocity	1		
28	0 MOTOR:minVel	TEMPLATE	0			RD		float	Motor[.Min_Velocity	1		
29	0 MOTOR:maxAcc	TEMPLATE	0			RD		float	Motor[.Max_Acceleration	1		
30	0 MOTOR:thrAcc	TEMPLATE	0			RD		float	Motor[.Acceleration_Threshold	1		
31	0 MOTOR:LwrLmtV	TEMPLATE	0			RD		float	Motor[.limitMinVelocity	1		
32	0 MOTOR:UprLmtV	TEMPLATE	0			RD		float	Motor[.limitMaxVelocity	1		
33	0 MOTOR:UprLmtA	TEMPLATE	0			RD		float	Motor[.limitAcceleration	1		
34	0 MOTOR:regCont	TEMPLATE	0			RD		Short	Motor[.regAry	59		
35	0 MOTOR:Error	TEMPLATE	0			RD		Short	Motor[.Error	1		
36	0 MOTOR:NumError	TEMPLATE	0					Short	Motor[.NumError	1		
37	0 MOTOR:dataldx	TEMPLATE	0			RD		Short	Motor[.dataIndex	1		
38	0 MOTOR:regStat	TEMPLATE	0			RD		byte	MotorReadReg[.motorStatus	1		
39	0 MOTOR:regPosi	TEMPLATE	0			RD		short	MotorReadReg[.motorPosition	1		
40	0 MOTOR:regExSta	TEMPLATE	0			RD		short	MotorReadReg[.motorExStatus	1		
41	0 MOTOR:mData	TEMPLATE	0			RD		short	Motor[1024		
42	1 Motor1	TWINCAT	1		1.0:1:131:169:9:232:1:1:<MOTOR>			Short				
43	2 Motor2	TWINCAT	1		2.0:801:131:169:9:232:1:1:<MOTOR>			Short				
44	40 istAry	TWINCAT	1		1.0:1:131:169:9:232:1:1	RD		long	.IstPosAry	32768		
45	51 datAry	TWINCAT	1		1.0:1:131:169:9:232:1:1	RD		long	.analogValueAry	32768		
46	52 selMotor	TWINCAT	1		1.0:1:131:169:9:232:1:1			short	.selMotor	1		

Pattern
!=1:04

CDI:Petra3 Motor Server continue...

TwinCAT PLC Control - KL2531.pro

File Edit Project Insert Extras Online Window Help

POUs

- FB_Motor_Init (FB)
- FB_Motor_Position (FB)
- FB_MotorFahren (FB)
- FB_Register_KL2531 (F)
- FB_Warte (FB)
- MAIN (PRG)

Global_Variables

```

0002  MotorReadReg[1]
0003      ...:motorStatus = 16#0F
0004      ...:motorPosition = 16#AD75
0005      ...:motorExStatus = 16#0000
0006  MotorReadReg[2]
0007  MotorWriteReg (%QB0)
0008  inCmdAlle
0009  Motor
0010      Motor[1]
0011          inCmd
0012          inRealCmd
0013              Status = 16#0002
0014              Error = 16#0000
0015              NumError = 16#0000
0016          regAry
0017              SollPositionAbs = 16#7FFD2694
0018              IstPositionAbs = 16#040EAD75
0019              IstHighWord = 16#040E
0020              IstLowWord = 16#AD75
0021              microSteps = 16#0040
0022              fullSteps = 16#00C8
0023              Rps = 4.651546
0024              Rms = 59539.8
0025              Rfs = 930.3093
0026              manualVelocity = 16#3CFF
0027              travelVelocity = 1674.557
0028              limitMinVelocity = 1.715828
0029              limitMaxVelocity = 3513.908
0030              limitAcceleration = 53617.98
0031              Min_Velocity = 1.716614
0032              Max_Velocity = 3300
0033              Max_Acceleration = 50003.28
0034              Acceleration_Threshold = 0.5358827
0035              rSollPosition = 6.039272e+007
0036              rIstPosition = 1914491
0037              stepSize = 16#0000FFFA
0038              dataIndex = 16#040F
0039          IstPosAry
0040          AnalogValueAry
0041  Motor[2]
    
```

Variable | Origin | Source

motorStatus	0x0F (15)	motorStatus, MotorReadReg[1], MotorReadReg, Inputs, Standard, KL2531
motorPosition	0xB76A (46954)	motorPosition, MotorReadReg[1], MotorReadReg, Inputs, Standard, KL2531
motorExStatus	0x0000 (0)	motorExStatus, MotorReadReg[1], MotorReadReg, Inputs, Standard, KL2531
motorCtrl	0x05 (5)	motorCtrl, MotorWriteReg[1], MotorWriteReg, Outputs, Standard, KL2531
motorVelocity	0x0000 (0)	motorVelocity, MotorWriteReg[1], MotorWriteReg, Outputs, Standard, KL2531

Instant Client (Instance 1)

Print ... Options ... Debug Tools ... Input Panel !

Device Context	TEST	Device Subsystem	ALL	<input type="checkbox"/> Show Stock Properties	
Device Server	P3MOTOR.CDI	Device Name	Motor1.CurPos	Device Property	RECV
Data Size	1	Data Type	LONG	Description	Issue CDI Read Telegram
Timeout					1000

/TEST/P3MOTOR.CDI/Motor1.CurPos RECV @ Jan 26 11:01:16.000

(0) 68003882

READ

STOP

Draw Mode

Text dun

Autoscale

Log Scale

Instant Client

Print ... Options ... Debug Tools ... Show Globals ! Input Panel !

Device Context	TEST	Device Subsystem	ALL	<input type="checkbox"/> Show Stock Properties	
Device Server	P3MOTOR	Device Name	Motor0	Device Property	Velocity
Data Size	1	Data Type	DOUBLE	Description	Motor Velocity

/TEST/P3MOTOR/Motor0 Velocity @ Jan 26 10:45:46.115

(0) 3300

Input Panel

Write Access

3300

ReferenceMove.STATUS

ReferenceMove.STOP

Reset

RotationMoveAllowed

RunCurrent

Status

StepCounter

Velocity

Draw Mode

Text dun

Autoscale

Log Scale

CDI: Current Status

- Bus Plugs for
 - SEDAC (SedPC, SedIP, SedUSB, SedISA) for Windows, Linux
 - CAN (CANOpen) for Windows, Linux
 - RS232 for Windows, Linux
 - TwinCat PLCs for Windows
 - Siemens Simatic PLCs for Windows
 - TICOM for Linux
- CDI loosely coupled to TINE
 - tine32.dll or libtine.so required
 - Can run in 'stand-alone' mode
- New Bus Plugs easy to create !
 - Independent of CDI Lib

System.exit(0)

CDI is plug-and-play.

Writing Bus Plugs is straightforward.

See Examples