Jacobsen Declaration Exhibit AK

Train Tools® Interface Programming in Visual Basic, Java and C/C++

Matt Katzer

KAM Industries
Portland, Or.

Agenda

- NMRA software application model
- Train Tools® Interface architecture
  - Key concepts and terms
  - Execution model
- Train Tools® Command set
- Writing an application (VB, Java, C/C++)
  - Using proposed NMRA API (Train Tools® interface) in VB
  - Using proposed NMRA API (Train Tools® interface) in C++
- Questions/Answers

Why are you here

- **Clinic will provide a status update on the NMRA software application model**
- **Clinic will review the TrainTools® API submitted to the NMRA DCC working group by KAM Industries.**
- **Clinic will focus on API architecture**
  - we will talk Application programming
  - API design tradeoffs
  - programming languages
  - implementation example programs (C++ and VB)
- **What are your expectations?**

Legal Disclaimer

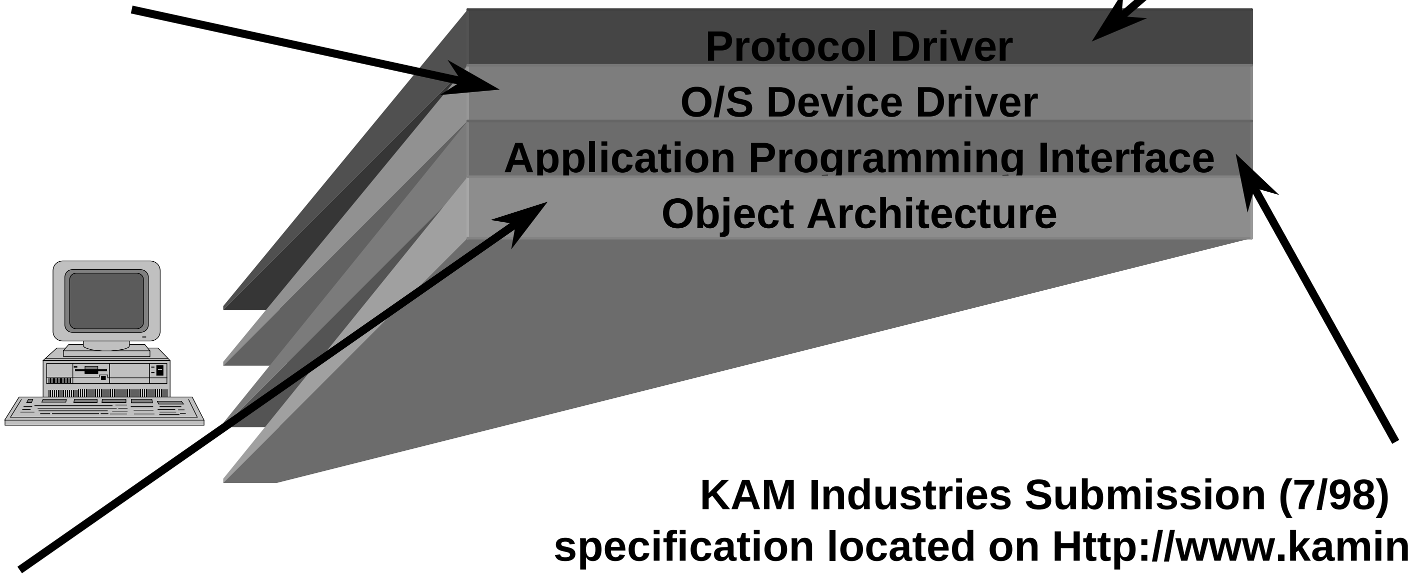
- KAM Industries has submitted the Trains Tools® Application Programming Interface to the NMRA DCC Working group for RP approval under the following conditions;
  - If the API is ratified as a Reference Practice(RP) KAM will transfer copyright of the document to the NMRA, otherwise the document and API's remain KAM's copyrighted property.
  - If the API is transferred to the NMRA, KAM retains rights to publish and use the RP Train Tools API document in their product, website and documentation as appropriate without any license fees or restrictions.

Status of NMRA Application S/W Architecture Model

- There are four parts to the NMRA DCC software architecture model

Katzer/Rice Draft Protocol Specification (7/9

no activity



Protocol Driver

O/S Device Driver

Application Programming Interface

Object Architecture

KAM Industries Submission (7/98)
specification located on Http://www.kamind.com

Rosa Proposal by Tannersoft (7/97)

Status of NMRA Application S/W Architecture Model (cont.)

- **Protocol Level**

  - **hardware Products**
    - » North Coast Engineering, Wangrow Electronics
    - » Easy DCC
    - » ZTC systems
  - **Software drivers for command station hardware**
    - » WinLok, Engine Commander®, Railroad Company Tayden Design
  - **Generic draft protocol driver**
    - » Engine Commander®

Status of NMRA Application S/W Architecture Model (cont.)

- **Device Driver Level**
  - » no activity

- **Application Interface Level**
  - – **hardware Products**
    - » not applicable to hardware
  - – **Microsoft COM/DCOM implementation of API**
    - » Engine Commander®
    - » Computer Dispatcher® (March 98)
    - » Generic type library available for linking with application written in Java, Visual Basic, C/C++
  - – **CORBA support**
    - » no activity

Status of NMRA Application S/W Architecture Model (cont.)

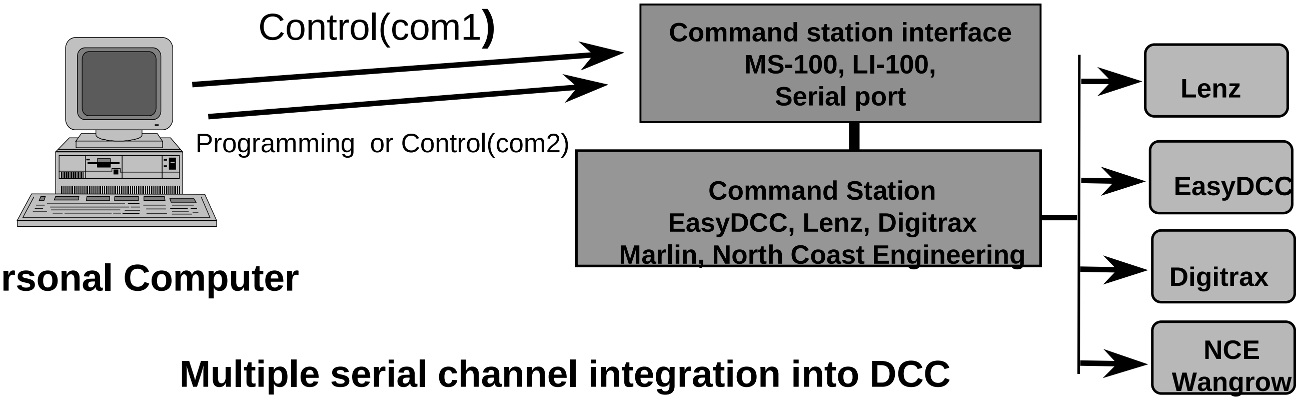
- **Object level**
  - **Rosa application model proposed (update on http://www.digi-toys.com)**
  - **hardware Products**
    - not applicable to hardware
  - **Software products**
    - Engine Commander® and Train Server® conforms in architecture model
  - **COM support**
    - no activity
  - **CORBA support**
    - no activity

Agenda

- NMRA software application model
- Train Tools® Interface architecture
  - Environment issues
  - Key concepts and terms
  - Execution model
- Train Tools® Command structure
- Writing an application (VB, Java, C/C++)
  - Using proposed NMRA API (Train Tools® interface) in VB
  - Using proposed NMRA API (Train Tools® interface) in C++
- Questions/Answers

Railroad Environment

- **Must have NMRA DCC compatible engines**
  - Pick a DCC supplier based on current required for your locomotive
  - By 2000, all locomotives in a price range above $100 will most likely have a decoder integrated into the unit
- **Command station equipment**
  - Expect a hybrid; plan for multiple command stations on layout
  - Model expected; one for programming the other for command and control



Multiple serial channel integration into DCC

Driving Force behind the API...

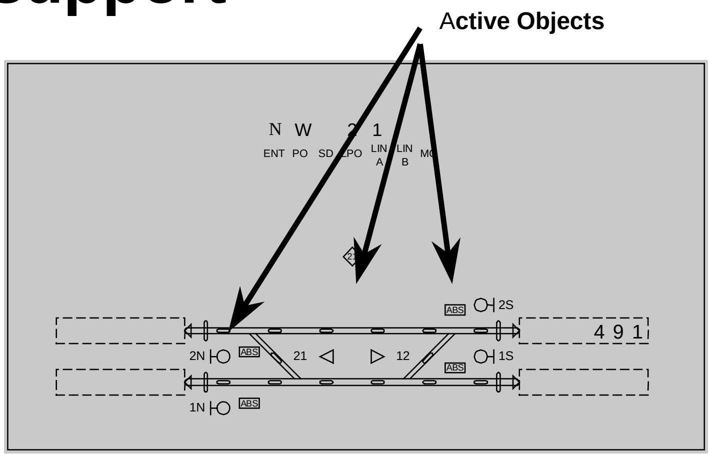
- **Train Tools® API was developed to address an internal needs at KAM**
  - **KAM needed away to control software costs and improve schedules**
    - non standard computer interfaces by command stations are costly to support
    - every one had their own architecture
  - **Standardization was needed to address product development**
    - API was required so KAM could decouple the GUI (client) from the backend (server) application
    - Needed to implement a Internet backbone
    - Needed a way to support Windows 95/98 and NT 4.0/5.0 distributed architecture
    - Needed an standard interface for a family of software products

Driving force (cont.)

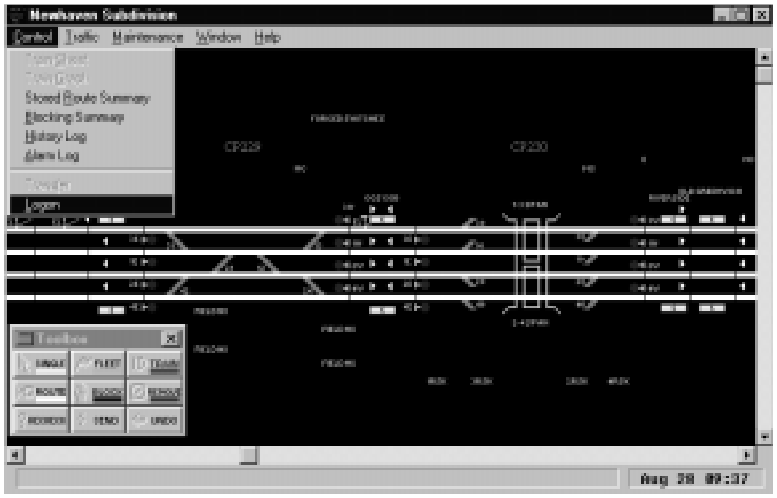
- **We needed an API that’s was language friendly**
  - Need flexibility to implement Java RNI if required
  - Needed support for Visual Basic
  - Needed support for C/C++
  - Needed support for our web servers via distributed Common Object Model (DCOM)
- **Our next generation software”Computer Dispatcher®” was an object driven model which required integrated network support.**
  - We needed an API that delivered functionality and implementation performance.
  - The API had to support COM and CORBA standards
  - The API ad to have source level compatibility at the minimum
- But the greatest factor for KAM was prototype control...

Computer Dispatcher®

- **The driving force for KAM was to build an infrastructure so we could support prototype operation….**



Active Objects



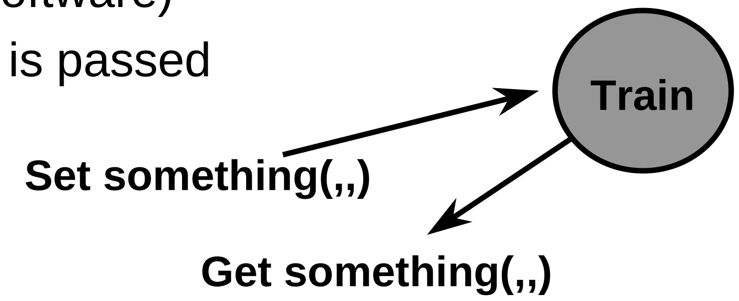
Computer Dispatchers
Model view of an active element with full Entry/Exit (route) control

CTC Panel View in Computer Dispatcher

Kansas City, Mo
NMRA 8/21/98

API Architecture

- **API is a combination of a property/method model; with an execution framework**
  - Objects are not passed in the API; rather states are passed
  - The state model reduces overhead on clients and improves the ability to port the API to different architecture (marshalling is expensive in software)
  - States are set; and execution is passed
    - » DccEngSetFunction(…..)
    - » DccEngGetSpeed(…)
    - » DccCommand(ObjectId)



Train

Set something(,,)

Get something(,,)

- **The API was designed to support prototype operations**

Architecture (cont.)

- **API is built on the following concepts**
  - **Devices are logical devices. There is a mapping between logical to physical**
    - » DccPortGetMaxLogPorts(lMaxLogical)
    - » PortGetMaxPhysical(lMaxPhysical, lMaxSerial, lMaxParallel)
    - » DccPortGetName(iComPort, strComPort)
    - » DccMiscGetControllerName(iController, strCntrl)
    - » DccPortSetConfig(iLogicalPort, 0, iPortRetrans, 0)
    - » DccPortSetMapController(iLogicalPort, iController, iPhysicalPort)
  - **Abstraction for the client was the key.**
    - » Client does not need configuration ability
    - » Client only needs to know how map a logical to a physical device
    - » The configuration extension was added to accommodate new manufactures equipment using a standard driver.

DCC Cooki e

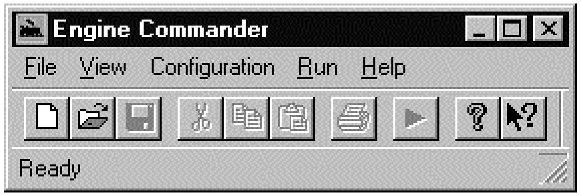
- **DCC addresses are integrated in an object**
  - – Objects have a reference and can be translated
  - – The object must be complete enough to use the API with as little information as possible
  - – Hence all information to control accessories or locomotives require and object as a reference
    - » This allows developers to implement the sever as an object store independent of the Operating System architecture.
    - » The objects then become a “DccCookie”.
      - DccCookie encapsulate programming ports, command ports, decoder class and DCC addresss
- The DCC Cookie becomes the reference token for system calls and can easily be validated

Architecture (cont.)

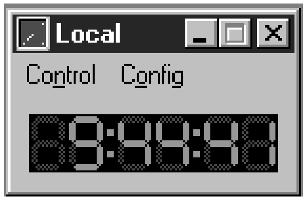
- **Abstraction also extends to decoders**
  - **we needed a model that allowed flexibility and growth**
    - » Decoder classes were created to group decoders.
    - » Each decoder class supports multiple decoder models
      - Classes are “Loco”, “Switch”,”Sensor”
      - Models are DH84, K87, LS110, Chub Detector1
    - » A set of decoder management fucntions were added to support applciaiton development
      - DccDecoderGetMaxModels(…)
      - DccDecoderGetModelName(…)
      - DccDecoderGetMaxAddress(…)
      - DccDecoderGetMfgName(…)
      - DccDecoderGetPowerMode(…)
      - DccDecoderGetModelFacility(…)
      - DccDecoderSetModelToObject(…)
  - **Objective was abstraction of the Interface**

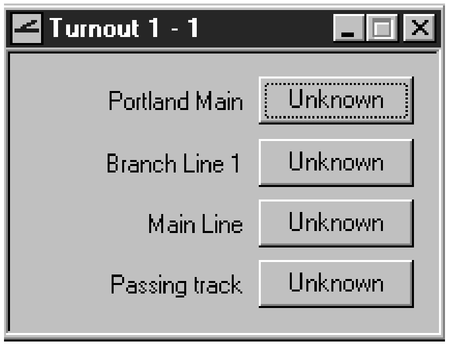
Engine Commander®

- **Built on a modular philosophy**
  - Implements all of the API's
  - Simple interface, but uses abstraction to reduce complexity of task
  - An accessory through switches..
  - A throttle run trains..
  - A clock tells time

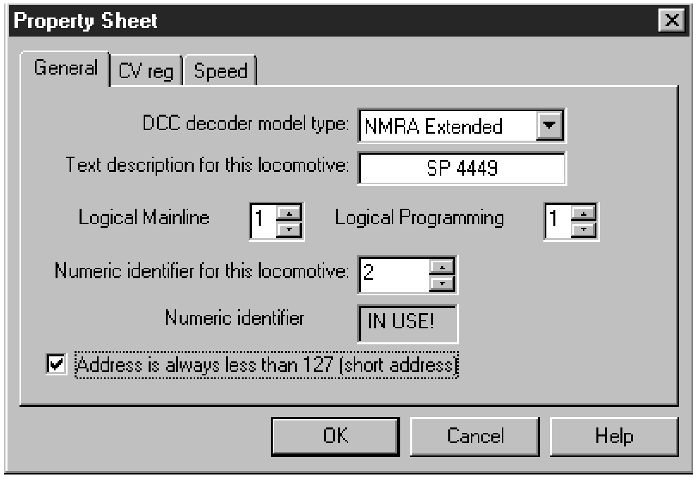

Engine Commander

Ready


Local


Turnout 1 - 1

Portland Main	Unknown
Branch Line 1	Unknown
Main Line	Unknown
Passing track	Unknown


Property Sheet

DCC decoder model type: NMRA Extended

Text description for this locomotive: SP 4449

Logical Mainline 1 Logical Programming 1

Numeric identifier for this locomotive: 2

Numeric identifier IN USE!

☑ Address is always less than 127 (short address)

Agenda

- NMRA software application model
- Train Tools® Interface architecture
  - Environment issues
  - Key concepts and terms
  - Execution model
- Train Tools® Command Summary
- Writing an application (VB, Java, C/C++)
  - Using proposed NMRA API (Train Tools® interface) in VB
  - Using proposed NMRA API (Train Tools® interface) in C++
- Questions/Answers

API command summary

- API Command classes
  - CV
  - Engine
  - Consist
  - Accessory
  - Command
  - Programming
  - Communications
  - Command
  - Decoder
  - Cab
  - Feedback
  - Callback methods

These are the major classes of commands needed in most DCC software applications.

We have implemented
Engine Commander®
and are in the development
phase of Computer Dispatcher®

•Train Tools API

- **Fucntions**
- **DccCVGetValue();**
  DccCVSetValue();
  DccCVGetStatus();
  DccCVSetStatus();
  DccCVGetName();
  DccCVGetMaxRegister();
  DccCVGetMinRegister();
- **Accessory Commands**
  DccAccGetFunction( );
  DccAccSetFunction( );
  DccAccGetFunctionAll( );
  DccAccSetFunctionAll( );
  DccAccGetFunctionMax( );
  DccAccGetName( );
  DccAccSetName( );
  DccAccGetFunctionName( );
  DccAccSetFunctionName( );

Train Tools API (cont.)

- **Engine**

  DccEngGetSpeed( );
  DccEngSetSpeed( );
  DccEngGetFunction( );
  DccEngSetFunction( );
  DccEngGetFunctionMax( );
  DccEngGetName( );
  DccEngSetName( );
  DccEngGetFunctionName( );
  DccEngSetFunctionName( );
  DccEngGetSpeedSteps( );
  DccEngSetSpeedSteps( );

- **Consist**

  DccEngConsistGetMax( );
  DccEngConsistSetParent( );
  DccEngConsistAddUnit( );
  DccEngConsistRemoveUnit( );
  DccEngConsistGetParent( );

Train Tools API(cont.)

- **Command Station**

  DccOprGetStationStatus();
  DccOprTurnOnStation();
  DccOprStartStation();
  DccOprClearStation();
  DccOprStopStation();
  DccOprPowerOn();
  DccOprPowerOff();
  DccOprHardReset();
  DccOprEmergencyStop();

- **Programming**

  DccProgramGetStatus();
  DccProgramSetMode( );
  DccProgramGetMode( );
  DccProgramWriteCV( );
  DccProgramReadCV( );
  DccProgramWriteDecoderToDataBase( );
  DccProgramReadDecoderFromDataBase( );

Train Tools API(cont.)

- **Communications**
  DccProgramGetStatus();
  DccProgramSetMode( );
  DccProgramGetMode( );
  DccProgramWriteCV( );
  DccProgramReadCV( );
  DccProgramWriteDecoderToDataBase( );
  DccProgramReadDecoderFromDataBase( );

- **Command**
  DccCmdCommand( );
  DccCmdConnect( );
  DccCmdDisConnect( );

- **Cab**
  DccCabWriteMessage( );
  DccCabReadMessage( );
  DccCabSetDccObject( );
  DccCabGetDccObject( );
  DccCabAdd( );
  DccCabDelete( );
  DccCabTranslate( );
  DccCabLookupDccObject( );

Train Tools API(cont.)

- **Decoder**

```
DccDecoderGetMaxModels();
DccDecoderGetModelName();
DccDecoderGetMaxAddress();
DccDecoderCheckAddrInUse();
DccDecoderGetMfgName(  );
DccDecoderGetPowerMode( );
DccDecoderAddAddr()
DccDecoderGetModelFacility()
DccDecoderReconnectObject( );
DccDecoderChangeAddress( )
DccDecoderTranslate( )
DccDecoderSetModelToObject()
DccDecoderGetMaxSpeed(  );
DccDecoderGetObjectCount()
DccDecoderGetObjectAtIndex()
DccDecoderDel( );
DccDecoderGetErrorState( )
```

Train Tools API(cont.)

- **Feedback**

```
DccFeedbackErrorMessage( );
DccFeedbackAccessoryBit( );
DccFeedbackAccessoryAll( );
DccFeedbackEngineResponse( );
DccFeedbackCV( );
DccFeedbackMessagesCab( );
DccFeedbackMisc( );
```

- **Callbacks**

```
DccResponseErrorMessage();
DccResponseAccessoryBit();
DccResponseAccessoryAll();
DccResponseEngineResponse();
DccResponseCV();
DccResponseCabMessage();
DccResponseMisc();
```

Train Tools Api(cont.)

- **Time**

```
DccMiscGetClockTime( );
DccMiscSetClockTime( );
```

- **Command Station**

```
DccMiscGetControllerName( );
DccMiscGetControllerNameAtPort( );
DccMiscGetCommandStationIndex( );
DccMiscMaxControllerID( );
DccMiscSetCommandStationValue( );
DccMiscGetCommandStationValue( );
DccMiscGetControllerFacility( );
```

- **Misc**

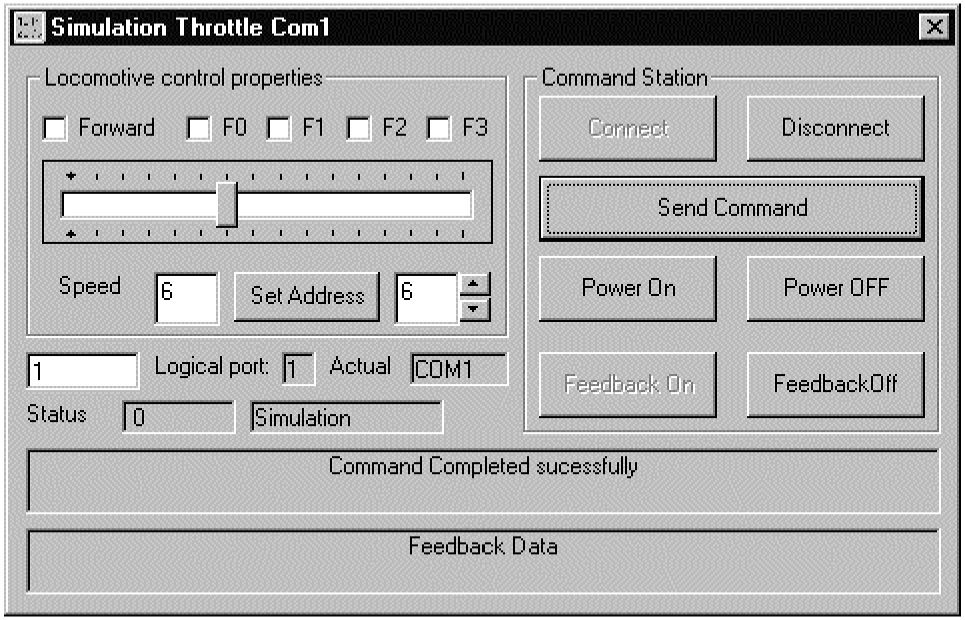
```
DccMiscGetErrorMsg ( );
DccMiscGetApiName( );
DccMiscGetInterfaceVersion( );
DccMiscSaveData( );
```

Agenda

- **NMRA software application model**
- **Train Tools® Interface architecture**
  - **Environment issues**
  - **Key concepts and terms**
  - **Execution model**
- **Train Tools® Command Summary**
- **Writing an application (VB, Java, C/C++)**
  - Using proposed NMRA API (Train Tools® interface) in VB
  - Using proposed NMRA API (Train Tools® interface) in C++
- **Questions/Answers**

Visual Basic Throttle?

- How is this Visual Basic application built?



Simulation Throttle Com1

Locomotive control properties

□ Forward □ F0 □ F1 □ F2 □ F3

Speed 6

Set Address 6

1 Logical port: 1 Actual COM1

Status 0 Simulation

Command Completed sucessfully

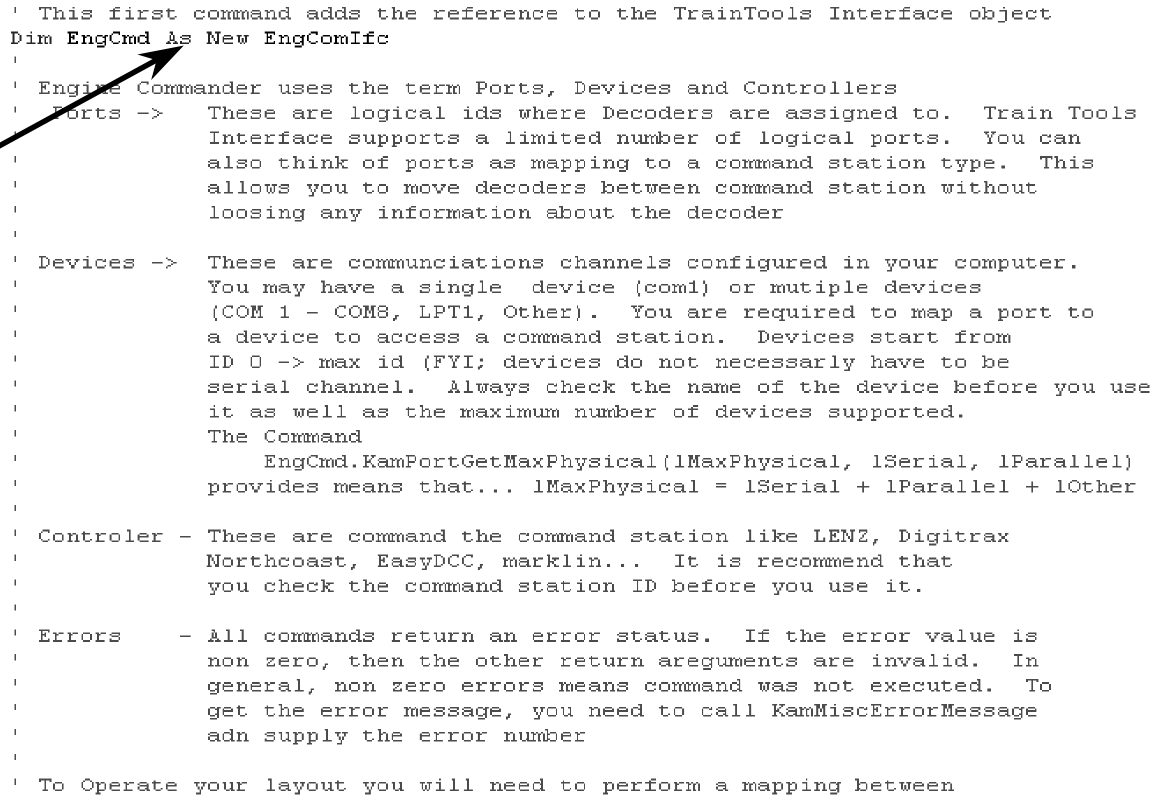
Feedback Data

- Lets look at how you program it

Visual Basic 5 Train Tools®

- **First step is to add the object reference**

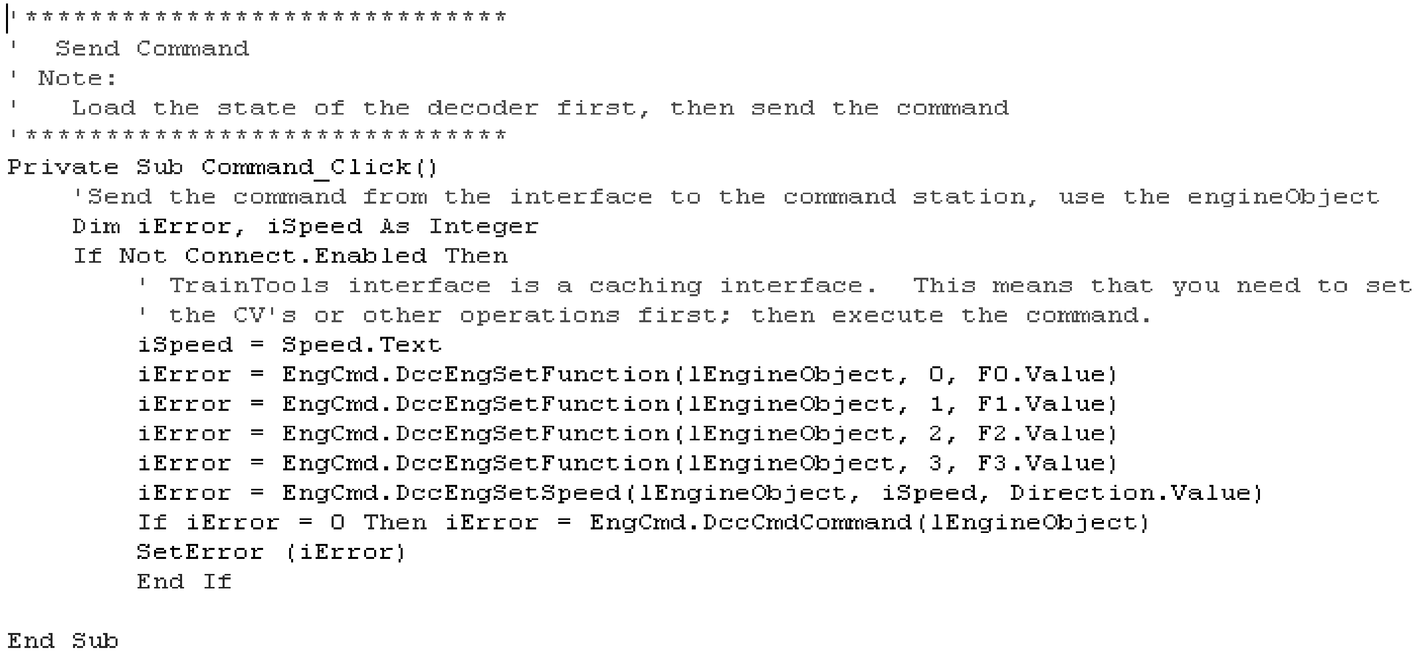
This is the key for all programming languages
We crete an object reference



```
' This first command adds the reference to the TrainTools Interface object
Dim EngCmd As New EngComIfc
'
' Engine Commander uses the term Ports, Devices and Controllers
'  Ports ->    These are logical ids where Decoders are assigned to.  Train Tools
'              Interface supports a limited number of logical ports.  You can
'              also think of ports as mapping to a command station type.  This
'              allows you to move decoders between command station without
'              loosing any information about the decoder
'
' Devices ->  These are communciations channels configured in your computer.
'              You may have a single  device (com1) or mutiple devices
'              (COM 1 - COM8, LPT1, Other).  You are required to map a port to
'              a device to access a command station.  Devices start from
'              ID 0 -> max id (FYI; devices do not necessarly have to be
'              serial channel.  Always check the name of the device before you use
'              it as well as the maximum number of devices supported.
'              The Command
'                  EngCmd.KamPortGetMaxPhysical(lMaxPhysical, lSerial, lParallel)
'              provides means that... lMaxPhysical = lSerial + lParallel + lOther
'
' Controler - These are command the command station like LENZ, Digitrax
'              Northcoast, EasyDCC, marklin...  It is recommend that
'              you check the command station ID before you use it.
'
' Errors    - All commands return an error status.  If the error value is
'              non zero, then the other return areguments are invalid.  In
'              general, non zero errors means command was not executed.  To
'              get the error message, you need to call KamMiscErrorMessage
'              adn supply the error number
'
' To Operate your layout you will need to perform a mapping between
```

Visual Basic 5 (cont.)

- **next,**
  - **Write the subroutine to control the loco**



```
'******************************
'  Send Command
' Note:
'   Load the state of the decoder first, then send the command
'******************************
Private Sub Command_Click()
    'Send the command from the interface to the command station, use the engineObject
    Dim iError, iSpeed As Integer
    If Not Connect.Enabled Then
        ' TrainTools interface is a caching interface.  This means that you need to set
        ' the CV's or other operations first; then execute the command.
        iSpeed = Speed.Text
        iError = EngCmd.DccEngSetFunction(lEngineObject, 0, F0.Value)
        iError = EngCmd.DccEngSetFunction(lEngineObject, 1, F1.Value)
        iError = EngCmd.DccEngSetFunction(lEngineObject, 2, F2.Value)
        iError = EngCmd.DccEngSetFunction(lEngineObject, 3, F3.Value)
        iError = EngCmd.DccEngSetSpeed(lEngineObject, iSpeed, Direction.Value)
        If iError = 0 Then iError = EngCmd.DccCmdCommand(lEngineObject)
        SetError (iError)
        End If

End Sub
```

Lets look at a C++ model

```
// Identify the interface of the object that we want to use...
MULTI_QI qi = {&IID_IEngComIfc, NULL, 0};
hr = CoCreateInstanceEx(CLSID_EngComIfc, NULL,
            CLSCTX_LOCAL_SERVER | CLSCTX_REMOTE_SERVER,
            pServerInfo, 1, &qi);
 //  add the security call at this point for compatibility for DCOM objects
 //CoInitializeSecurity
// Now make the com conenction for the interface
     if (SUCCEEDED(qi.hr))
     {
     // Now get the remote TrainTools interface
     short sError;
     m_pEngIfc = (IEngComIfc*)qi.pItf;
     GetVersion(&m_csIfcVersion );
     m_pEngIfc->DccPortGetMaxLogPorts(&m_iMaxLogicalPorts, &sError);
     m_pEngIfc->DccPortGetMaxPhysical(&m_iMaxPhysicalPorts, &m_iMaxSerialPorts, &m_iMaxParallelPorts, &sError );
     m_pEngIfc->DccMiscMaxControllerID(&m_iMaxControlerId, &sError);
```

This is the key for all programming languages
We crete an object reference

A little more complex, but very similar to VB

C++ cont.

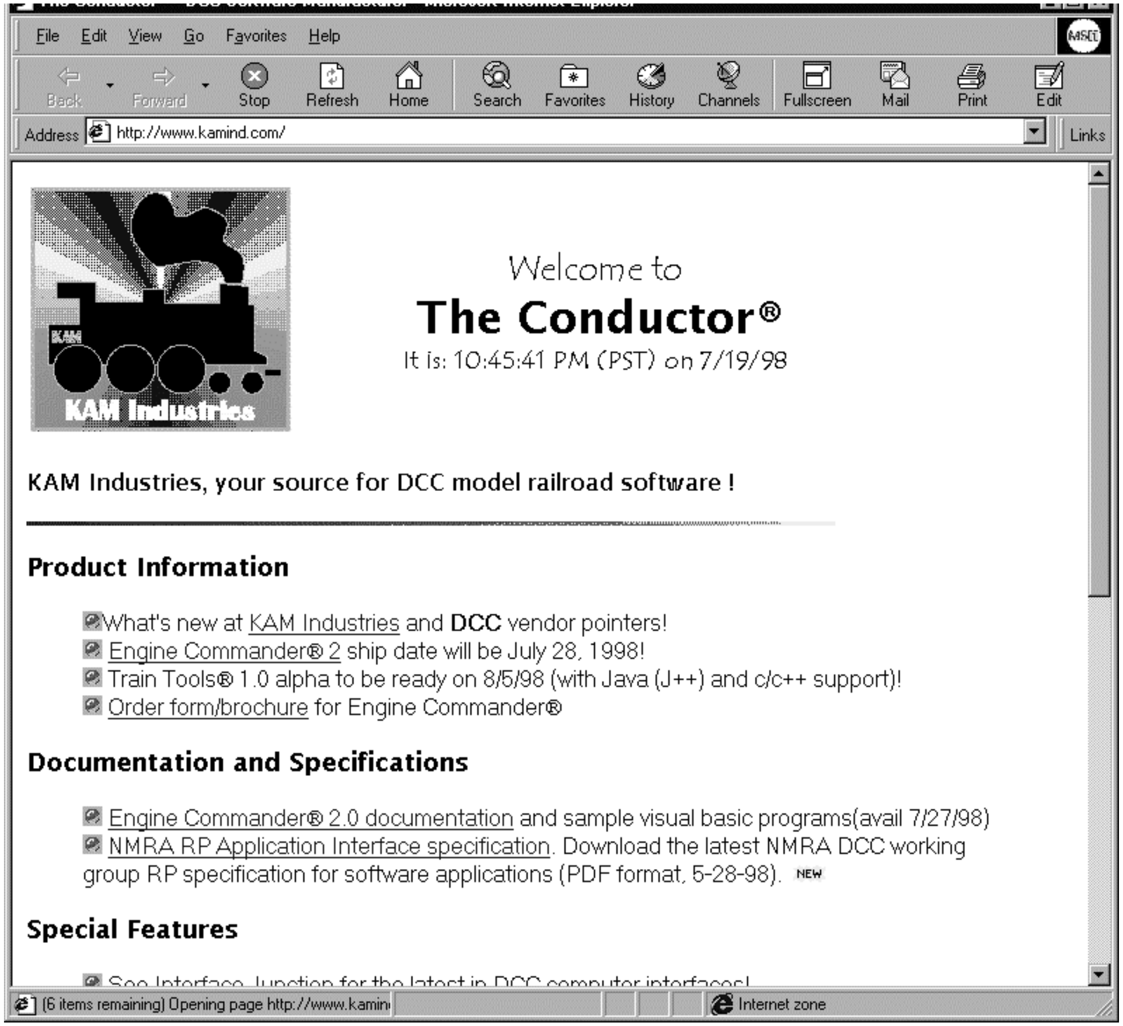
```
/*
 *     NAME
*                  DecoderGetModelFromCookie() - Get controller facilities.
*      RETURN VALUE
*                  iModel - Decoder model ID.
 *
 *     DESCRIPTION
*                  DecoderGetModelFromCookie() gets the decoder model ID.
 */

int  TInterfaceDevice::DecoderGetModelFromCookie(long     lCookie ) const
{
        TRACE( "TInterfaceDevice::DecoderGetModelFromCookie( 0x%08lx ) - Entering\n",lCookie );
        short iError;
         int iLogCmdPort, iLogProgPort, iDCCAddr, iDecoderClass, iDecoderModel;
        m_pEngIfc->DccDecoderTranslate(lCookie, &iLogCmdPort, &iLogProgPort, &iDCCAddr,
                                                                      &iDecoderClass, &iDecoderModel, &iError);
        TRACE( "TInterfaceDevice::DecoderGetModelFromCookie( 0x%08lx ) - Exiting: (%X)- Error\n", lCookie, iError );
        return ( iDecoderModel );
}
```

Easily supported in multiple languages

Where to from here?

- Download the API from our web page
- Visit KAM at the Train show and pick up a free demo CD (beta product) (booth 240-250)
- The Train Tools® API is real
  - EngineCommander is designed around it
  - Computer Dispatcher development is in process
- Sends us your feedback to
  - TrainTools@kam.rain.com
  - We want to hear your suggestions and recommendations

http://www.kamind.com


The Conductor — DCC Software Manufacturer - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back
Forward
Stop
Refresh
Home
Search
Favorites
History
Channels
Fullscreen
Mail
Print
Edit
Address
http://www.kamind.com/
Links
KAM
KAM Industries
(6 items remaining) Opening page http://www.kamin
Internet zone

Welcome to

The Conductor®

It is: 10:45:41 PM (PST) on 7/19/98

KAM Industries, your source for DCC model railroad software !

Product Information

- What's new at KAM Industries and **DCC** vendor pointers!
- Engine Commander® 2 ship date will be July 28, 1998!
- Train Tools® 1.0 alpha to be ready on 8/5/98 (with Java (J++) and c/c++ support)!
- Order form/brochure for Engine Commander®

Documentation and Specifications

- Engine Commander® 2.0 documentation and sample visual basic programs(avail 7/27/98)
- NMRA RP Application Interface specification. Download the latest NMRA DCC working group RP specification for software applications (PDF format, 5-28-98). NEW

Special Features

- See Interface Junction for the latest in DCC computer interfaces!

Questions ?

Matt Katzer
email: mkatzer@kam.rain.com
web: http://kam.rain.com
home: 503-291-1221