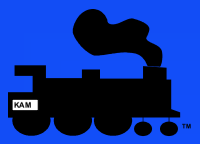
Model Railroad Computer Control

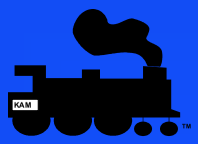
(How I am going to write my Train Program)

Matt Katzer
Portland, Or.



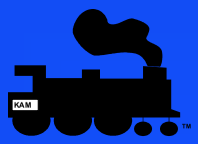
Agenda

- Philosophy
- System Requirements
  - Hardware, Software, Interfaces and Objects
- How should I write software
- TrainTools™ example
  - Marklin (AC/DC) example
  - NMRA DCC example
  - VisualBasic 5.0 object example
- Key Web pages you will need to visit
- EngineCommander™ and VisualBasic Demo



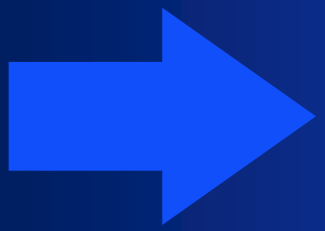
Why are you here

- **Clinic will focus on writing programs to control your railroad….**
  - we will talk about PC's
  - programming languages and objects
  - example programs in VisualBasic
- **Visual Basic to TrainTools demostration**
- **What are your expectations?**



Philosophy

- **Computer Controlled**
  - The computer controls the routes of the train
  - The operator runs his/her layout from the computer
- **Computer Monitored**
  - **The computer is a tool of the modeler**
  - **The computer is used to manage event**
  - **The computer does not control!**



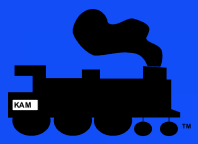
I like to write software, but I want to run trains and use computers monitor the layout

Hardware Requirements

- **What type of PC hardware should you buy?**
  - Intel Pentium or Pentium II (buy the fastest you can get! 200+ with MMX)
  - Cache 512K+ and 32 - 64 Mbytes of system memory (win-98 you want lots)
  - 1Gbyte-2 Gbyte Mbyte (SCSI) hard disk
  - 3.5” Floppy drive and CDROM Drive (SCSI)
  - Sound Blaster Pro (SCSI version)
  - VGA graphics card (with 1024 x 1280 support)
  - VGA color monitor with SVGA support (17” or 21“)
  - Must be able to run win-95/98 or win-NT 4.0
  - Intelligent serial card! (if you want more then 2 communications ports)
- **What you should not buy**
  - 286, 386 or 486 PC’s (ROT: must run Win 95)
  - systems that contain less then 16 Mbytes
- **You must have internet access!**

Software Requirements

- **Software Required to develop applications**
  - Do you want to use Borland or Microsoft tools
  - What language do you use (don't use Pascal)
    - » Visual Basic 5.0
    - » C/C++ (Microsoft or others)
    - » Java 1.1/1.2 (Microsoft or Sun)
    - » COM (Common Object Module) enabled?
  - Do you want to buy object components (like TrainTools™ or develop your own.
- **Other tools**
  - Microsoft's developer library (MSDN)
  - Third part debugging tools

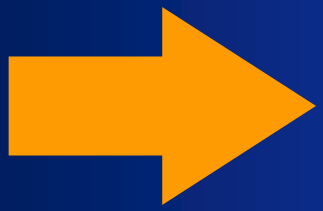


Railroad Requirements

- Must have NMRA DCC compatible Decoders
- Need to select Manufactures Interface
  - NorthCoast, EasyDCC and ZTC supports NMRA TP spec
  - Digitrax; proprietary serial Interface (needs 16.4K baud)
  - Lenz requires LI-100 interface
  - All interfaces are on http://kam.rain.com
- Where to get interface specs
  - http://kam.rain.com look under infaceterface
  - NMRA specs are at http://www.nmra.org
- Where to get Interface Objects
  - KAM's EngineCommander™ with TrainTools™ interface
  - TrainTools™ Train objects(to be in alpha later this year)

What's an object

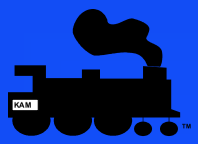
- Ok, whats this object thing, and why does everyone talk about it?
  - Objects are a way to describe something; an object contains characteristics, properties and functions
  - a cup; is ceramic and is either can be filled, or empty
  - There are two object models that describe this -- Microsoft's COM or CORBA (Dec/IBM/Sun driven)
  - There's a software war and it take 18-24 months to resolve



So what does this mean to me?

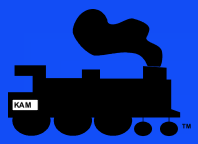
So what does this mean?

- You should not worry about it..
  - But make sure the vendors know your requirements
    - » distributed multiple clients and servers
    - » network enabled (you should be able to use the internet and connect layouts)
    - » VB, Java, C++ language support
  - Pick your programming language (let the object suppliers worry about the interface)
- You need to look for a set of Application Interfaces (API's) that you can use
  - Language independent (Java, C++ and Visual Basic)
  - Vendor agrees to extend them to future standard
  - Command station independence



Agenda

- Philosophy
- System Requirements
  - Hardware, Software, Interfaces and Objects
- How should I write software
- TrainTools™ example
  - Marklin (AC/DC) example
  - NMRA DCC example
  - VisualBasic 5.0 object example
- Key Web pages you will need to visit
- EngineCommander™ and VisualBasic Demo



Command Control Software

- **Four classes of products available**
  - Mac & Apple
  - MS-DOS
  - Microsoft Windows (16 bit mode 95 and win-NT)
  - Windows-95, Windows-98 and Windows NT 4.x/5.x
- **Traditional Supplied programs**
  - not object based
  - scripting language included
  - supporting old 16 bit environments
- **Next generation**
  - Object based, supporting Microsoft COM and DCOM
  - published API's available for all languages

Should I write my own programs?

- Are you ready to
  - read the protocol specification to the controller
  - write in a computer language
  - spend many hours away from you layout
  - -have fun programming?

- What Language do you use?
  - novice: basic or visual basic
  - experienced: C or Java (J++)
  - advanced: C++ under windows

Understand where you want to put your energy to maximize your fun!

What is the best way to begin?

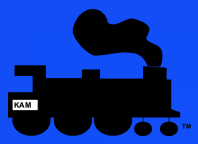
- First understand the protocol and interface
- Second follow these rules
  - keep it simple…
  - design the architecture…
  - build the infrastructure...
- Best way to begin..
  - buy the correct PC and the Microsoft tools
  - if you are a novice used Visual Basic
  - if you are advance user, use Visual C++

Remember, small steps are better then big ones!

What is the best way to begin?

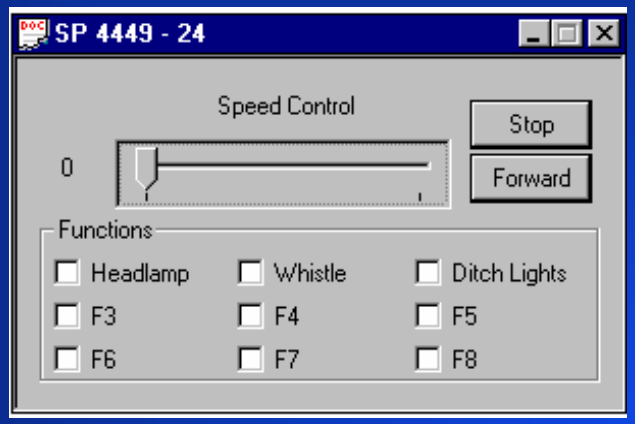
- **Follow these four steps**
  1. Acquire a 3rd party app for experimentation
  2. Look at the traditional application
     - You need to decide onusing objects or tradional methods. We will look at both
  2. Design your user interface (use GUI tool)
  4. Now implement small features
- **Lets walk through these four steps...**

Remember: Used objects when possible!



Acquire a 3rd party Application for testing

- **Do you want to make this or program it..**



- **Lets look at how you program it**
  1. Review Interface Spec
  2. Review Sample apps

Traditional Visual Basic example

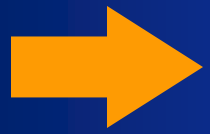

```
Sub cmdGo_Click ()
    '--- If the port is openned,
    If MSComm1.PortOpen Then
       '--- Send the command to the port
       MSComm1.Output = Chr$(wEngine) + Chr$(wSpeed)
       End If

End Sub
```

Tradational C example (6050)

```
// The format of the data packet is
// Format:  <engine Address> <direction/Speed>
//     Where <Engine Address>   is between 1 - 99
//                              (6050 internface does not transmit address 80 -99)
//           <speed/direction>  b7 = 0
//                              b6 = 1
//                              b5 = Direction 0 for forward, 1 for reverse
//                              b4 = function set to 1 if on, 0 if off
//                              b3 - 1 = speed
case ENGINE: {
    BYTE bySpeedDir;
    WORD wEngine;
    LPWORD lpwEng;
    // Get the engine number form the bit sequence
    lpwEng = (LPWORD) (lpSrcData + ENGADDR_PKT);     // set the engine peed location
    wEngine = *lpwEng;     // get the engine number
    bySpeedDir = lpSrcData[ENGSPEED_PKT] & 0x0F;     // get the speed; 0 - 0xF are speeds
    // Process direction; in this case speed of 0x0F is reverse to the controller
    if (lpSrcData[ENGDIRECT_PKT]) bySpeedDir = 0x0F; // set to reverse speed
    // Process function
    if (lpSrcData[ENGFUNC_PKT]) bySpeedDir = bySpeedDir | 0x10;  // set function on
          else                  bySpeedDir = bySpeedDir & 0xEF;  // set function off (mask to 0)
    lpDstData[0] = bySpeedDir;      // Set the speed and engine function
    lpDstData[1] = (BYTE) wEngine;  // set the engine address
    iDataSize = 2;}
    break;
```

Problem with Traditional methods

- **Controller dependent operation**
  - If you want to move to a new command station, you need to rewrite the interface
  - Can migrate easily to new technologies.
  - Can't support multiple command stations or divisions without major software changes (man years of effort!)
- **What you want to do…**
  - Command station independence
  - Communications device independence
  - Divisional support
  - Network distribution (client server support)

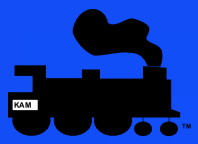


Lets look at an Object specification

Visual Basic Example

Demo!!!

- **Ok what did we see**
  - **TrainTools™ interface supplied with EnigneCommander™**
  - **We used the local server, instead of a networked server**
  - **Ran a Visual Basic Application**
- **Lets look at how this was done!**



Review Appplications Interface Specification (IDL)

KamEngGetSpeed

Parameter List		Type	Range	Direction	Description
iDCCAddr	int	[1]	In	Decoder address	
lpSpeed	int *	[2]	Out	Pointer to locomotive speed	
lpDirection	int *	[3]	Out	Pointer to locomotive direction	

[1] 1-127 for short addresses, 1-10239 for long addresses.

[2] Speed range is dependent on whether the decoder is set to 14,18, or 128 speed steps and matches the values defined by NMRA S9.2 and RP 9.2.1. 0 is stop and 1 is emergency stop for all modes.

[3] Forward is boolean `TRUE` and reverse is boolean `FALSE`.

Return Value	Type	Range	Description
iError short	[1]	Error flag	

[1] *iError* = 0 for success. Nonzero is an error number (see **KamMiscGetErrorMsg**).

KamEngGetSpeed takes the decoder address and pointers to locations to store the locomotive speed and direction as parameters. It sets the memory pointed to by *lpSpeed* to the locomotive speed and the memory pointed to by *lpDirection* to the locomotive direction.

Visual Basic 4/5 Object Example

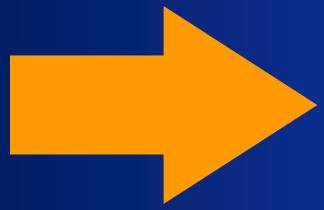
```
' This first command adds the reference to the TrainTools Interface object to Visual Basic
 Dim EngCmd As New EngComIfc

 ' Connect the DCC address to the Object
 Private Sub Command_Click()
    'Send the command from the interface to the command station
    Dim iStatus, iAddr, iSpeed As Integer
    If Not Connect.Enabled Then
       ' TrainTools interface is a caching interface.  This means that you need to set up the CV's
      ‘ or other operations first; then ' execute the command.
       iAddr = Address.Text
       iSpeed = Speed.Text
       iStatus = EngCmd.KamEngPutFunction(iAddr, 0, F0.Value)
       iStatus = EngCmd.KamEngPutFunction(iAddr, 1, F1.Value)
       iStatus = EngCmd.KamEngPutFunction(iAddr, 2, F2.Value)
       iStatus = EngCmd.KamEngPutFunction(iAddr, 3, F3.Value)
       iStatus = EngCmd.KamEngPutSpeed(iAddr, iSpeed, Direction.Value)
       If iStatus = 0 Then iStatus = EngCmd.KamCmdCommand(iAddr)
       SetError (iStatus)
       End If

 End Sub
```

Visual C++ Object Example

```
//*************************************************************************************
//  Execute Command to an address
//*************************************************************************************
// Command -> Command the decoder
int TInterfaceDevice::Command(TDecoderDevice* pDevice) // returns an interface string
      {
      short iResults;
      TEngineDevice *pEngine = (TEngineDevice *)pDevice;
       m_pEngIfc->KamEngPutFunction(pEngine->m_iDCCAddress, 0, pEngine->m_bFLState, &iResults);
       m_pEngIfc->KamEngPutFunction(pEngine->m_iDCCAddress, 1, pEngine->m_bF1State, &iResults);
       m_pEngIfc->KamEngPutFunction(pEngine->m_iDCCAddress, 2, pEngine->m_bF2State, &iResults);
       m_pEngIfc->KamEngPutFunction(pEngine->m_iDCCAddress, 3, pEngine->m_bF3State, &iResults);
       m_pEngIfc->KamEngPutFunction(pEngine->m_iDCCAddress, 4, pEngine->m_bF4State, &iResults);
       m_pEngIfc->KamEngPutFunction(pEngine->m_iDCCAddress, 5, pEngine->m_bF5State, &iResults);
       m_pEngIfc->KamEngPutFunction(pEngine->m_iDCCAddress, 6, pEngine->m_bF6State, &iResults);
       m_pEngIfc->KamEngPutFunction(pEngine->m_iDCCAddress, 7, pEngine->m_bF7State, &iResults);
       m_pEngIfc->KamEngPutFunction(pEngine->m_iDCCAddress, 8, pEngine->m_bF8State, &iResults);
       m_pEngIfc->KamEngPutSpeed(pEngine->m_iDCCAddress, pEngine->m_iOldSpeed,
                                          pEngine->m_bForward, &iResults);
       m_pEngIfc->KamCmdCommand(pDevice->m_iDCCAddress, &iResults);
      }
return(iResults);
   }
```

Design your GUI Interface

- Pick your GUI development tool
  - Visual C++
  - Visual J++
  - Visual Basic 5.0



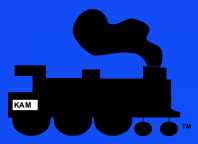
Just drop in an Object and use the interface!

Bottom Line..
Have fun

Lets Look at where to get free software!

Shareware Software and Support

- You must get on the internet
  - a lot of software is free, but it is on the net
  - all support is structured to be downloaded from web pages
- Lets look at some methods
  - Compuserver
  - KAM and John_Kabat home pages
  - other web pages
- Key web page for software…
  - http://www.microsoft.com



http://kam.rain.com

http://ourworld.compuserve.com/homepages/John_Kabat/

Other DCC web pages..

DCC Hardware

http://www.lenz.com

http://www.digitrax.com

http://www.wangrow.com

http://www.tttrains.com/tttrains/dccdiv.htm

DCC Software:

http://kam.rain.com

http:// ourworld.compuserve.com/homepages/John_Kabat/

DCC information

http://www.tttrains.com/dcc/

http://www.mcs.net:80/~weyand/nmra/

http://www.mcs.net/~dsdawdy/NMRA/dcc.html

http://www.tttrains.com/tttrains/

Questions ?

Now, lets run Trains!

Matt Katzer
email: mkatzer@kam.rain.com
web: http://kam.rain.com
home: 503-291-9879

EngineCommander™

- **Built on a modular philosophy**
  - users can add complexity at their own pace
  - simulation interface is include to help other programmers
- **Good user documentation**
  - Interface specification available with product
  - Supports all popular command stations
- **Allows local execution or server execution**
  - Design to support up to 10 command station divisions
  - Language independence

EngineCommander™ Modes

- **Serial Interfaces supported**
  - Marklin 6050/6023 interfaces on AC and DC boosters
  - Lenz LI-100
  - Digitrax, Wangrow serial Interface
  - NMRA serial interface
  - EasyDCC
- **Key features**
  - Mobile and stationary decoder support (command and programming mode)
  - Local or remote execution
  - Visual basic interface supplied (run with ours or roll your own)