Sequence of calls involved in one log entry.

Manually worked out by reading the code.

Flash => sim.exe => CNCPlotterDialog.dll.

immersive\_log MILL PLOTTER.txt

**13** <43436> **= >Flash FS Command Call> -** <invoke name=**"editor\_group"** returntype=**"xml"**><arguments><string>**set\_active\_program setActiveProgramCallback O30007**</string></arguments></invoke>**13** <43436> **= >Flash FS Command Call> -** <invoke name=**"editor\_group"** returntype=**"xml"**><arguments><string>**set\_active\_program setActiveProgramCallback O30007**</string></arguments></invoke>

**Flash**

biz\immersive\ControllerModule\HaasController\LatheV16\screens\ProgramEditScreen.as

**function** DisplaySelectedProgram**(** **):void**

**{**

**var** ret**:**String **=** ""**;**

**this.**SetDefaultMode**();**

**this.**\_program\_list\_View**.**SelectCurrentHighlitedProgram**();**

ret **=** **this.**\_program\_list\_View**.**GetCurrentHighlitedProgram**();**

**this.**NCcontroller**.**operator\_panel**.**program\_list**.**SetCurrentProgram **(** ret **);**

**this.**MakeProgramListInvisible**();**

**this.**SetFonts**();**

**this.**UpdateProgramDisplay**();**

**this.**SetCurrentProgram**();**

**}**

biz\immersive\ControllerModule\OperatorModule\ProgModule\ProgramList.as

public **function** SetCurrentProgram**(** progr\_name**:**String **):void**

**{**

**this.**SetCurrentControllerProgram**(** progr\_name **);**

\_ctrl**.**sim\_com**.**setActiveProgramCall**(** **this.**current\_program\_name **);**

**}**

biz\immersive\ControllerModule\EICallbackClass.as

public **function** setActiveProgramCall**(** programName**:**String **):void**

**{**

**if(** local\_mode **)**

**{**

**this.**setActiveProgramCallback**(** programName**,** "0" **);**

**return;**

**}**

**var** callConstruct**:**String **=** "set\_active\_program setActiveProgramCallback " **+** programName**;**

**this.**MakeExternalCall**(** "editor\_group"**,** callConstruct**);**

**}**

**C++ (sim.exe)**

SIM\_MAIN/flash\_syntax.h

#define SET\_ACTIVE\_PROGRAM\_EDITOR\_GROUP "set\_active\_program"

SIM\_MAIN\FlexVCBridgeDlg.cpp

// Receive events from Flash:

BEGIN\_EVENTSINK\_MAP**(**FlexVCBridgeDlg**,** CDialog**)**

ON\_EVENT**(**FlexVCBridgeDlg**,** IDC\_SHOCKWAVEFLASH\_NEW**,** 197**,** FlexVCBridgeDlg**::**FlashCallShockwaveflash1**,** VTS\_BSTR**)**

END\_EVENTSINK\_MAP**()**

void FlexVCBridgeDlg**::**FlashCallShockwaveflash1**(**LPCTSTR request**)**

**{**

ProcessFlashCall**(** request **);**

**}**

void FlexVCBridgeDlg**::**ProcessFlashCall**(**LPCTSTR request**)**

**{**

CMarkup xml**;**

// Non blocking call

aw **=** **new** ExternalFlashThreadCom**(** action\_string**,** arg\_string **);**

**}**

SIM\_MAIN\ExternalFlashThreadCom.cpp

void ExternalFlashThreadCom**::**execute**(**void**)**

**{**

process\_ie\_call**(** **this->**ActionString**,** **this->**ArgString **);**

**}**

void process\_ie\_call**(** CString AString**,** CString Arg **)**

**{**

**if(** AString**.**CompareNoCase**(** EDITOR\_GROUP **)** **==** 0 **)**

**{**

**if(** Arg**.**GetLength**()** **!=** 0 **)**

ie\_process\_editor\_group**(** Arg **,** **true** **);**

**return;**

**}**

SIM\_MAIN\ie\_call\_editor\_group.cpp

void ie\_process\_editor\_group**(**CString string\_to\_parse**,** int callback\_flag**)**

**{**

**if** **(**temp\_str**.**CompareNoCase**(**\_T**(**SET\_ACTIVE\_PROGRAM\_EDITOR\_GROUP**))** **==** 0**)** **{** // process flash requests to load a file on the local hard drive

ie\_editor\_group\_set\_active\_prgm**(**next\_args**,** callback\_flag**);**

**}**

**}**

void ie\_editor\_group\_set\_active\_prgm**(** CString string\_to\_parse **,** int callback\_flag**)**

**{**

prg\_name **=** string\_to\_parse**.**Tokenize**(**" "**,**curPos**);**

global\_p\_MainController**->**set\_current\_nc\_program**(** prg\_name **);**

**}**

SIM\_MAIN\easyrob\_controller.cpp

void easyrob\_controller**::**set\_current\_nc\_program**(** CString prog\_name **)**

**{**

**this->**current\_nc\_program **=** **this->**TheProgramList**->**set\_current\_program**(** prog\_name **);**

**if(** **this->**current\_nc\_program **==** **NULL** **)**

**return;**

**this->**current\_nc\_program**->**update\_plotter**();**

**return;**

**}**

SIM\_MAIN\program\_list.cpp

program\_storage **\***program\_list**::**set\_current\_program**(** CString new\_program **)**

**{**

program\_storage **\***aprogram**;**

char prg\_name**[**256**];**

strcpy**(** prg\_name **,** **(**LPCTSTR**)** new\_program **);**

aprogram **=** program\_list**::**find\_program**(** prg\_name **);**

**if(** aprogram **==** **NULL** **)**

**return** aprogram**;**

program\_list**::**current\_active\_prg **=** aprogram**;**

**return** aprogram**;**

**};**

SIM\_MAIN\program\_storage.cpp

int program\_storage**::**update\_plotter **(** void **)**

**{**

int ret **=** 0**;**

global\_p\_MainController**->**m\_jobQ**->**addJob**(** **new** PreviewNCProgram**(** **this** **)** **);**

**return** ret**;**

**}**

PreviewNCProgram**::**PreviewNCProgram**(** program\_storage **\***a\_prg **)**

**{**

**this->**the\_prg **=** a\_prg**;**

**}**

void PreviewNCProgram**::**execute**(**void**)**

**{**

**if(** **this->**the\_prg **==** **NULL** **)**

**return;**

int ret **=** **this->**the\_prg**->**update\_plotter\_thread**();**

**}**

int program\_storage**::**update\_plotter\_thread**(**void**)**

**{**

int ret **=** 0**;**

**if** **(**global\_p\_MainController **==** **NULL)**

**return** ret**;**

CArray**<**const char**\*>** c**;**

CritSectEx**::**Scope scope**;** // empty constructor doesn't lock anything

scope**.**Lock**(**program\_cs**);**

**this->**UpdateProgramPositions**();**

// Program Program preview

// Modified Created

// {\_program\_modified) (dirty\_flag) What to do

//

// 0 TRUE Create Program Preview Send Program

// 1 TRUE Create Program Preview Send Program

// 0 FALSE Just send Program

// 1 FALSE Create Program Preview Send Program

**if** **(this->**\_program\_modified **==** 1 **||** **this->**preview\_path**.**GetNumBlocks**()** **==** 0**)** **{** // Create Preview

**this->**preview\_path**.**ClearList**();**

ret **=** global\_p\_MainController**->**create\_program\_preview\_path**(this);**

**this->**preview\_path**.**dirty\_flag **=** **true;**

**this->**\_program\_modified **=** 0**;**

ret **=** **this->**SendPreviewPathTOPlotter**();** // Send Program

**this->**preview\_path**.**dirty\_flag **=** **false;**

**}**

**else** **{** // just set new program

**if** **(this->**preview\_path**.**GetNumBlocks**()** **!=** 0**)** **{**

CncPlotterDialog\_SetProgramPath**(**global\_p\_MainController**->**machine\_name**.**GetBuffer**(),** **(** immersive**::**CncProgramPath**\*** **)** **&this->**preview\_path**);**

**}**

**}**

**if** **(**global\_p\_MainController**->**PreviewCompletedCallback**.**GetLength**()** **!=** 0**)** **{**

c**.**Add**(**ret **?** "1" **:** "0"**);**

API\_CallBlockingFlashFunction**(**global\_p\_MainController**->**PreviewCompletedCallback**.**GetBuffer**(),** c**);**

**}**

scope**.**Unlock**();** // We are done Unlock

**return** ret**;**

**}**

int program\_storage::SendPreviewPathTOPlotter ( void )

{

global\_p\_MainController->motion\_engine.UpdatePreviewWorkOffsets();

global\_p\_MainController->motion\_engine.UpdatePreviewToolOffsets();

if( this->preview\_path.GetNumBlocks() != 0 )

{

CncPlotterDialog\_SetProgramPathWorkToolOffset( global\_p\_MainController->machine\_name.GetBuffer() ,

(immersive::CncProgramPath \*)&this->preview\_path ,

&(global\_p\_MainController->motion\_engine.preview\_workoffset\_list) ,

&(global\_p\_MainController->motion\_engine.preview\_tooloffset\_list) );

}

}

**C++ (CNCPlotterDialog.dll)**

CNCPlotterDialog\CncPlotterDialog.cpp

CNC\_PLOTTER\_DIALOG\_API void CncPlotterDialog\_SetProgramPathWorkToolOffset( const std::string& machine, CncProgramPath\* path, WorkOffsetList\* work\_offsets,ToolGeometryFileList\* tools)

{

AFX\_MANAGE\_STATE( AfxGetStaticModuleState() );

CCncPlotterDialogDlg\* dialog = theApp.GetCncPlotterDialogWindow();

if ( dialog )

dialog->SetProgramPathWorkToolOffset( machine, path, work\_offsets, tools );

}

void CCncPlotterDialogDlg::SetProgramPathWorkToolOffset( const std::string& machine, immersive::CncProgramPath\* path, immersive::WorkOffsetList\* work\_offsets, immersive::ToolGeometryFileList\* tools )

{

if ( this->mOSG )

this->mOSG->SetProgramPathWorkToolOffset( machine, path, work\_offsets, tools );

}

CNC\_PLOTTER\_DIALOG\_API void CncPlotterDialog\_SetProgramPath( const std::string& machine, CncProgramPath\* path )

{

AFX\_MANAGE\_STATE( AfxGetStaticModuleState() );

CCncPlotterDialogDlg\* dialog = theApp.GetCncPlotterDialogWindow();

if ( dialog )

{

dialog->SetProgramPath( machine, path );

}

}

CNCPlotterDialog\CncPlotterDialogDlg.h

void SetProgramPath( const std::string& machine, immersive::CncProgramPath\* path ) { if ( this->mOSG ) this->mOSG->SetProgramPath( machine, path ); }

CNCPlotterDialog\MFC\_OSG.h

typedef enum {

ZERO\_EVENT = 0,

STOP\_PROGRAM,

START\_PROGRAM,

START\_PROGRAM\_NAME,

UPDATE\_FINAL\_PART,

UPDATE\_TOOL\_LIST,

UPDATE\_WORKOFFSET\_LIST,

UPDATE\_PATH\_WORK\_TOOL\_LIST,

UPDATE\_CNC\_PATH,

HIDE\_FINAL\_PART,

RESET\_EVENT,

WINDOW\_SIZING\_EVENT,

CLEAR\_SLIDE\_BAR\_TIMER\_EVENT,

CLEAR\_ACTIVE\_PROGRAM\_EVENT,

MAX\_NUMBER\_OF\_EVENTS

} AnimationEventType;

CNCPlotterDialog\MFC\_OSG.cpp

void cOSG::SetProgramPathWorkToolOffset( const std::string& machine, immersive::CncProgramPath\* path, immersive::WorkOffsetList\* work\_offsets, immersive::ToolGeometryFileList\* tools )

{

this->\_current\_machine = machine;

this->\_current\_path = path;

this->\_work\_offsets = work\_offsets;

this->\_tools = tools;

EventItem a\_new\_event;

a\_new\_event.SetEventType( UPDATE\_PATH\_WORK\_TOOL\_LIST );

this->\_event\_mgr.PushEvent( a\_new\_event );

}

void cOSG::SetProgramPath( const std::string& machine, immersive::CncProgramPath\* path )

{

this->\_current\_machine = machine;

this->\_current\_path = path;

EventItem a\_new\_event;

a\_new\_event.SetEventType( UPDATE\_CNC\_PATH );

this->\_event\_mgr.PushEvent( a\_new\_event );

}

void cOSG::PreFrameUpdate()

{

EventItem a\_event;

int t1 = 0;

int t2 = 0;

int t3 = 0;

int t4 = 0;

// Due any preframe updates in this routine

while( \_event\_mgr.AnyEventsToProcess() )

{

a\_event = this->\_event\_mgr.PopEvent();

switch( a\_event.GetEventType () )

{

case UPDATE\_PATH\_WORK\_TOOL\_LIST:

this->\_scene.SetWorkOffsets( this->\_machine, this->\_work\_offsets );

this->\_scene.SetToolGeometryFileList( this->\_machine, this->\_tools );

if( this->\_current\_path != NULL )

{

this->\_scene.ProgramReset();

this->\_scene.SetProgramPath( this->\_current\_machine, this->\_current\_path );

this->\_scene.SetCurrentProgramFinalPartDisplay( this->\_enable\_final\_flag );

this->ResetActiveProgramView();

}

break;

case UPDATE\_CNC\_PATH:

if( this->\_current\_path != NULL )

{

this->\_scene.ProgramReset();

this->\_scene.SetProgramPath( this->\_current\_machine, this->\_current\_path );

this->\_scene.SetCurrentProgramFinalPartDisplay( this->\_enable\_final\_flag );

this->ResetActiveProgramView();

}

break;

}

CNCPlotterDialog\SceneManager.cpp

void SceneManager::SetProgramPath( const std::string& machine, immersive::CncProgramPath\* path )

{

this->\_previewPath = \*path;

this->grepProgramPath();

}

void SceneManager::grepProgramPath( void )

{

this->\_progMgr.AddProgram( this->\_previewPath );

this->ProgramReset();

}

void SceneManager::ProgramReset( void )

{

osg::Vec3 pos( 0.0f, 0.0f, 0.0f );

osg::Vec3 workOffset( 0.0f, 0.0f, 0.0f );

int toolId( 0 );

this->ProgramStop();

this->\_progMgr.ResetActiveProgramStepCounter( pos, workOffset, toolId );

this->\_toolMgr.MoveTool( pos );

this->SetUpdate();

this->\_tc->MoveToTime( 0.0f );

}