MTConnect Constant Value Mapping Notes

Monday, August 26, 2019

This is the output from the Okuma Lathe.

2019-08-23T16:38:57.5937500Z|avail|AVAILABLE|S1Mode|SPINDLE|C5Mode|INDEX|estop|ARMED|pmode|MANUAL|pprogram|MIT-CLAMP-EXT1.min|pexecution|STOPPED|p1block||p1line|UNAVAILABLE|p2block||p2line|UNAVAILABLE|S1speed|0|S1cmd|0|S1load|0|C5actm|193.797|C5load|0|X1actm|24848.553|X1load|35|Z1actm|20111.054|Z1load|5|Z2actm|2672.821|Z2load|3|Z4actm|10479.6|Z4load|UNAVAILABLE|Z5actm|10479.6|pFovr|0|p1Fact|UNAVAILABLE|p1Fcmd|0|p1LPathPos|476.547 0 727.684|p2Fact|UNAVAILABLE|p2Fcmd|0|p2LPathPos|0 0 1143.028

2019-08-23T16:38:57.5937500Z|system|NORMAL||||

Of note most of the tag renames are straightforward. p1Fact|UNAVAILABLE is a scary value, but it exist. HOWEVER, the tag path\_feedrateovr does not exist so can't be remapped into a XML tag mapping. So we need to make a constant so the value exists. Note, the Okuma.txt (which contains a device description) for the MTConnect probe, has a field for path\_feedrateovr even though the Okuma SHDR will never output/produce a value for this field.

Here are the tag renames, including the new CONSTANTS section:

[TAGRENAMES]

X1actm=Xabs

Z1actm=Zabs

# NO

S1speed=Srpm

## There are two spindles

pexecution=execution

Fovr=path\_feedrateovr

p1Fact=path\_feedratefrt

pFovr=path\_feedrateovr

pprogram=program

pmode=controllermode

###########################################################

[ENUMREMAPPING]

# Okuma lathe

pexecution.READY=IDLE

pexecution.ACTIVE=EXECUTING

pexecution.INTERRUPTED=PAUSED

pexecution.STOPPED=PAUSED

[CONSTANTS]

path\_feedrateovr=100

The is a new declaration in adapter.cpp

static std::map<std::string, std::string> valuemapping;

static std::map<std::string , std::map<std::string, std::string> > constantdevmapping;

that enumerates all the constants PER device. So the first map template name is the Device name, and the second template name (a map) is a pair: tag name, constant value.

void Adapter::CheckAlias(Device \*device, std::string &key, std::string &value)

{

//////////////////////////////

// Change enumerations

if(enummapping.find(key+"."+value)!= enummapping.end())

{

value=enummapping[key+"."+value];

}

// Map shdr key (e.g., mode) into new key (controllermode)

if(keymapping.find(key)!= keymapping.end())

{

key=keymapping[key];

}

}

In the master file (MTConnectAgentFromShdr.cpp ) , upon reading the Okuma.ini file it generates a new config.ini file:

// this is based on the type to define a section FOR ONE DEVICE NOT ALL

std::map<std::string, std::string> consts=ini.GetMap("CONSTANTS");

if(consts.size()>0)

{

cfg.MergeKeys(StdStringFormat("%s", \_devices[j].c\_str())+".CONSTANTS", consts);

order.push\_back(StdStringFormat("%s", \_devices[j].c\_str())+".CONSTANTS");

}

Later the specific Config.ini file generated is parsed again to read all the generated renames.

This code generates a valuemapping for a specific device.

// clear all device named value remappings

Adapter::constantdevmapping.clear();

// get a vector of all DEVICE.CONSTANTS sections

std::vector<std::string> constantsections = cfg. GetSectionBranch (".CONSTANTS");

// For each device constant section

for(size\_t j=0; j< constantsections.size(); j++)

{

// Search for the Device name that preceeds the .

size\_t fnd = constantsections[j].find(".");

// If not found, this should be an error, but we just continue

if(fnd == std::string::npos)

continue; // should not be

// Get the device name.

std::string devicename=constantsections[j].substr(0,fnd);

// Get the constants (tap/const value) map associated with this device

std::map<std::string, std::string> constassigns= cfg.GetMap(constantsections[j]);

// Assign this map to this device

Adapter::constantdevmapping[devicename]= constassigns;

}

During processing of the SHDR every time a new tag/value pair is read not only is the tag/value checked for a rename or enumeration rename, but a constant value is emitted.

// Handle constants if any for this device

void Adapter::CheckConstants(Device \*device, std::string shdrtime, std::string &key, std::string &value)

{

// Handle constants if any for this device - EVERY TIME

if(device && constantdevmapping[device->getName()].size() >0)

{

std::map<std::string, std::string > consts = constantdevmapping[device->getName()];

size\_t cnt=0;

for(std::map<std::string, std::string >::iterator it= consts.begin(); it!= consts.end(); it++)

{

DataItem \*dataItem;

std::string constkey = (\*it).first;

if(constkey==key)

{

// remap constant value to key's value

value = toUpperCase((\*it).second);

cnt++;

}

else

{

dataItem = device->getDeviceDataItem(constkey);

dataItem->setDataSource(this);

mAgent->addToBuffer(dataItem, toUpperCase((\*it).second), shdrtime);

// check to see if this is the same key as the shdr key if no give it the constant value

}

}

}

}

FIXME: make sure that the const value catches any tag rename and eliminates them or subsitutues the value.