minimal.cps

This is an example of a fairly minimal yet still meaningful post configuration script.

```
/**
  Copyright (C) 2012-2014 by Autodesk, Inc.
  All rights reserved.

  Minimal post processor configuration.

  $Revision$
  $Date$

  FORKID {96F3CC76-19C0-4828-BF27-6A50AED3B187}
*/

description = "Minimal Heidenhain";
vendor = "Autodesk";
vendorUrl = "http://www.autodesk.com";
legal = "Copyright (C) 2012-2014 by Autodesk, Inc.";
certificationLevel = 2;

longDescription = "Minimal milling post for Heidenhain.";

extension = "h";
setCodePage("ansi");

capabilities = CAPABILITY_MILLING;

var spindleAxisTable = new Table(["X", "Y", "Z"], {force:true});

var radiusCompensationTable = new Table(
  [" R0", " RL", " RR"],
  {initial:RADIUS_COMPENSATION_OFF},
  "Invalid radius compensation"
);

var xyzFormat = createFormat({decimals:(unit == MM ? 3 : 4), forceSign:true});
var feedFormat = createFormat({decimals:(unit == MM ? 0 : 2), scale:(unit == MM ? 1 : 10)});
var rpmFormat = createFormat({decimals:0});
var mFormat = createFormat({prefix:"M", decimals:0});


var xOutput = createVariable({prefix:" X"}, xyzFormat);
var yOutput = createVariable({prefix:" Y"}, xyzFormat);
var zOutput = createVariable({prefix:" Z"}, xyzFormat);
var feedOutput = createVariable({prefix:" F"}, feedFormat);

var blockNumber = 0;

/**
  Writes the specified block.
*/
function writeBlock(block) {
  writeln(blockNumber + SP + block);
  ++blockNumber;
}

function onOpen() {
  writeBlock(
    "BEGIN PGM" + (programName ? (SP + programName) : "") + ((unit == MM) ? " MM" : " INCH")
  );
  writeBlock(mFormat.format(3)); // spindle on - clockwise

  machineConfiguration.setRetractPlane(-1.0); // safe machine retract plane (M91)
}

/**
  Invalidates the current position and feedrate. Invoke this function to
  force X, Y, Z, and F in the following block.
*/
function invalidate() {
  xOutput.reset();
  yOutput.reset();
  zOutput.reset();
  feedOutput.reset();
}

function onSection() {
  writeBlock("L Z" + xyzFormat.format(machineConfiguration.getRetractPlane()) + " M91");
  var retracted = true;

  writeBlock(
    "TOOL CALL " + tool.number + SP + spindleAxisTable.lookup(spindleAxis) + " S" + rpmFormat.format(tool.spindleRPM)
  );

  setTranslation(currentSection.workOrigin);
  setRotation(currentSection.workPlane);
```

```
  invalidate();

  var initialPosition = getFramePosition(currentSection.getInitialPosition());
  if (!retracted) {
    if (getCurrentPosition().z < initialPosition.z) {
      writeBlock("L" + zOutput.format(initialPosition.z) + " FMAX");
    }
  }
  writeBlock("L" + xOutput.format(initialPosition.x) + yOutput.format(initialPosition.y) + zOutput.format(initialPosition.z));
}

function onRapid(x, y, z) {
  var xyz = xOutput.format(x) + yOutput.format(y) + zOutput.format(z);
  if (xyz) {
    writeBlock("L" + xyz + radiusCompensationTable.lookup(radiusCompensation) + " FMAX");
    feedOutput.reset();
  }
}

function onLinear(x, y, z, feed) {
  var xyz = xOutput.format(x) + yOutput.format(y) + zOutput.format(z);
  var f = feedOutput.format(feed);
  if (xyz) {
    writeBlock("L" + xyz + radiusCompensationTable.lookup(radiusCompensation) + f);
  }
}

function onSectionEnd() {
  // full retract in machine coordinate system
  writeBlock("L Z" + xyzFormat.format(machineConfiguration.getRetractPlane()) + " R0 FMAX " + mFormat.format(91));
  invalidate();
}

function onClose() {
  writeBlock(mFormat.format(30)); // stop program, spindle stop, coolant off
  writeBlock(
    "END PGM" + (programName ? (SP + programName) : "") + ((unit == MM) ? " MM" : " INCH")
  );
}
```

Generated by Autodesk, Inc. 10 August 2017
Copyright (c) 2012-2017 by Autodesk, Inc.