

```

package net.minecraft.src;

import java.io.*;

// Created by MoareAI
// Contains data handled by the ModLoader

public class mod_MiscDigital extends BaseMod
{
    //ModLoader actions
    public mod_MiscDigital()
    {
        RenderId1 = ModLoader.getUniqueBlockModelID(this, false);
        RenderGate = RenderId1;
        ModLoader.getLoadedMods();
        ModLoader.RegisterBlock(BlockToggle);
        ModLoader.RegisterBlock(BlockRS);
        ModLoader.RegisterBlock(BlockClock);
        ModLoader.RegisterBlock(BlockHalfAdder);
        ModLoader.RegisterBlock(BlockJKFlipFlop);
        ModLoader.RegisterBlock(BlockPulseGen);
        ModLoader.RegisterTileEntity(net.minecraft.src.TileEntityPulseClock.class,
"PulseClock");
        ModLoader.RegisterTileEntity(net.minecraft.src.TileEntityPulseGen.class,
"PulseGenerator");

        //Block Texture
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/MDToggleOn.png",
TextureToggleOn);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/MDToggleOff.png",
TextureToggleOff);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/MDSRON.png",
TextureSRON);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/MDRSOn.png",
TextureRSOn);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/MDRSOff.png",
TextureRSOff);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/MDClockOn.png",
TextureClockOn);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/MDClockOff.png",
TextureClockOff);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/MDHalfAdder.png",
TextureHalfAdder);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/MDJKOn.png",
TextureJKOn);
        ModLoader.addOverride("/terrain.png", "/MoareAI/Blocks/MDJKOff.png",
TextureJKOff);

        //Item Icons
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/MDToggle.png",
IconToggle);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/MDRS.png", IconRS);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/MDClock.png",
IconClock);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/MDHalfAdder.png",
IconHalfAdder);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/MDJK.png", IconJK);
        ModLoader.addOverride("/gui/items.png", "/MoareAI/Items/MDPulseGen.png",
IconPulseGen);

        //Item names
        ModLoader.AddName(ItemToggle, "Toggle");
        ModLoader.AddName(ItemRS, "RS Latch");
        ModLoader.AddName(ItemClock, "Pulse Clock");
    }
}

```

```

mod_MiscDigital.java

ModLoader.AddName(ItemHalfAdder, "Half Adder");
ModLoader.AddName(ItemJKFlipFlop, "JK Flip Flop");
ModLoader.AddName(ItemPulseGen, "Pulse Generator");

//Block Names
ModLoader.AddName(BlockToggle, "Toggle");
ModLoader.AddName(BlockRS, "RS Latch");
ModLoader.AddName(BlockClock, "Pulse Clock");
ModLoader.AddName(BlockHalfAdder, "Half Adder");
ModLoader.AddName(BlockJKFlipFlop, "JK Flip Flop");
ModLoader.AddName(BlockPulseGen, "Pulse Generator");

if (EffectiveCrafting)
    CraftCost = 1;
else
    CraftCost = 0;

//Recipe for Toggle
ModLoader.AddRecipe(new ItemStack(ItemToggle, CraftCost*5+1), new Object[] {
    " Z ", "#XY", Character.valueOf('#'), Item.redstone,
    Character.valueOf('X'), Block.sand, Character.valueOf('Y'), Block.torchRedstoneActive,
    Character.valueOf('Z'), Block.lever
});

//Recipe for RS
ModLoader.AddRecipe(new ItemStack(ItemRS, CraftCost*2+1), new Object[] {
    "XY#", "# #", "#YX", Character.valueOf('#'), Item.redstone,
    Character.valueOf('X'), Block.sand, Character.valueOf('Y'), Block.torchRedstoneActive
});

//Recipe for Pulse Clock
ModLoader.AddRecipe(new ItemStack(ItemClock, CraftCost*5+1), new Object[] {
    "#XY", Character.valueOf('#'), Item.pocketSundial, Character.valueOf('X'),
    Block.sand, Character.valueOf('Y'), Block.torchRedstoneActive
});

//Recipe for Half Adder
ModLoader.AddRecipe(new ItemStack(ItemHalfAdder, 1), new Object[] {
    "#", "X", Character.valueOf('#'), mod_LogicalGates.ItemXOR,
    Character.valueOf('X'), mod_LogicalGates.ItemAND
});

//Recipe for JK Flip Flop
ModLoader.AddRecipe(new ItemStack(ItemJKFlipFlop, 1), new Object[] {
    "#X", Character.valueOf('#'), ItemRS, Character.valueOf('X'), ItemToggle
});

//Recipe for Pulse Generator
ModLoader.AddRecipe(new ItemStack(ItemPulseGen, 1), new Object[] {
    " #", "YX ", Character.valueOf('#'), mod_LogicalGates.ItemAND,
    Character.valueOf('X'), mod_LogicalGates.ItemNOT, Character.valueOf('Y'),
    Item.redstoneRepeater
});
ModLoader.AddRecipe(new ItemStack(ItemPulseGen, 1), new Object[] {
    " #", "X ", Character.valueOf('#'), ItemRS, Character.valueOf('X'),
    Item.redstoneRepeater
});
}

public String Version()
{
    return "1.6.5";
}

```

```

//Block declaration
public static final Block BlockRS;
public static final Block BlockClock;
public static final Block BlockToggle;
public static final Block BlockHalfAdder;
public static final Block BlockJKFlipFlop;
public static final Block BlockPulseGen;

//Render type declaration
int RenderId1;
public static int RenderGate = 201;

//Texture declaration
public static int TextureToggleOn;
public static int TextureToggleOff;
public static int TextureSROn;
public static int TextureRSOn;
public static int TextureRSOff;
public static int TextureClockOn;
public static int TextureClockOff;
public static int TextureHalfAdder;
public static int TextureJKOn;
public static int TextureJKOff;

//Icon declaration
public static int IconToggle;
public static int IconRS;
public static int IconClock;
public static int IconHalfAdder;
public static int IconJK;
public static int IconPulseGen;

//Item declaration
public static Item ItemToggle;
public static Item ItemRS;
public static Item ItemClock;
public static Item ItemHalfAdder;
public static Item ItemJKFlipFlop;
public static Item ItemPulseGen;

//Properties
public static final File cfgfile;
public static int PulseGenLength = 2;
public static int ClockTime = 16;
public static boolean EffectiveCrafting = true;
private int CraftCost;
public static boolean ReduceIcon = false;
public static boolean ReduceTexture = true;
public static int GUIStep = 5;
public static int BlockToggleID = 207;
public static int BlockRSID = 208;
public static int BlockClockID = 209;
public static int BlockHalfAdderID = 210;
public static int BlockJKFlipFlopID = 211;
public static int BlockPulseGenID = 212;
public static int ItemToggleID = 463;
public static int ItemRSID = 464;
public static int ItemClockID = 465;
public static int ItemHalfAdderID = 466;
public static int ItemJKFlipFlopID = 467;
public static int ItemPulseGenID = 468;

//Block and Item data
static

```

```

{
    //The configuration file
    cfgfile = new File(Minecraft.getMinecraftDir(), "/config/MoareAI's Misc Digital
Functions.cfg");
    try
    {
        cfgfile.getParentFile().mkdirs();
        if(cfgfile.exists() || cfgfile.createNewFile())
        {
            Properties properties = new Properties();
            if(cfgfile.canRead())
            {
                FileInputStream fileinputstream = new FileInputStream(cfgfile);
                properties.load(fileinputstream);
                System.out.println("Succesfully loaded Proterties for MoareAI's
Misc Digital Functions");
                PulseGenLength =
Integer.parseInt(properties.getProperty("DefaultPulseLength", "2"));
                ClockTime =
Integer.parseInt(properties.getProperty("DefaultClockPulseWidth", "16"));
                EffectiveCrafting =
Boolean.parseBoolean(properties.getProperty("EffectiveCrafting", "true"));
                ReduceTexture =
Boolean.parseBoolean(properties.getProperty("ReduceBlockSprites", "false"));
                ReduceIcon =
Boolean.parseBoolean(properties.getProperty("ReduceItemSprites", "false"));
                GUIStep = Integer.parseInt(properties.getProperty("GUIStep", "5"));
                BlockToggleID =
Integer.parseInt(properties.getProperty("BlockToggleID", "207"));
                BlockRSID =
Integer.parseInt(properties.getProperty("BlockRSLatchID", "208"));
                BlockClockID =
Integer.parseInt(properties.getProperty("BlockPulseClockID", "209"));
                BlockHalfAdderID =
Integer.parseInt(properties.getProperty("BlockHalfAdderID", "210"));
                BlockJKFlipFlopID =
Integer.parseInt(properties.getProperty("BlockJKFlipFlopID", "211"));
                BlockPulseGenID =
Integer.parseInt(properties.getProperty("BlockPulseGeneratorID", "212"));
                ItemToggleID =
Integer.parseInt(properties.getProperty("ItemToggleID", "463"));
                ItemRSID = Integer.parseInt(properties.getProperty("ItemRSLatchID",
"464"));
                ItemClockID =
Integer.parseInt(properties.getProperty("ItemPulseClockID", "465"));
                ItemHalfAdderID =
Integer.parseInt(properties.getProperty("ItemHalfAdderID", "466"));
                ItemJKFlipFlopID =
Integer.parseInt(properties.getProperty("ItemJKFlipFlopID", "467"));
                ItemPulseGenID =
Integer.parseInt(properties.getProperty("ItemPulseGeneratorID", "468"));
                fileinputstream.close();
            }
            if(cfgfile.canWrite())
            {
                FileOutputStream fileoutputstream = new FileOutputStream(cfgfile);
                properties.setProperty("DefaultPulseLength",
Integer.toString(PulseGenLength));
                properties.setProperty("DefaultClockPulseWidth",
Integer.toString(ClockTime));
                properties.setProperty("EffectiveCrafting",
Boolean.toString(EffectiveCrafting));
                properties.setProperty("ReduceBlockSprites",
Boolean.toString(ReduceTexture));
            }
        }
    }
}

```

```

        properties.setProperty("ReduceItemSprites",
Boolean.toString(ReduceIcon));
        properties.setProperty("GUIStep", Integer.toString(GUIStep));
        properties.setProperty("BlockToggleID",
Integer.toString(BlockToggleID));
        properties.setProperty("BlockRSLatchID",
Integer.toString(BlockRSID));
        properties.setProperty("BlockPulseClockID",
Integer.toString(BlockClockID));
        properties.setProperty("BlockHalfAdderID",
Integer.toString(BlockHalfAdderID));
        properties.setProperty("BlockJKFlipFlopID",
Integer.toString(BlockJKFlipFlopID));
        properties.setProperty("BlockPulseGeneratorID",
Integer.toString(BlockPulseGenID));
        properties.setProperty("ItemToggleID",
Integer.toString(ItemToggleID));
        properties.setProperty("ItemRSLatchID",
Integer.toString(ItemRSID));
        properties.setProperty("ItemPulseClockID",
Integer.toString(ItemClockID));
        properties.setProperty("ItemHalfAdderID",
Integer.toString(ItemHalfAdderID));
        properties.setProperty("ItemJKFlipFlopID",
Integer.toString(ItemJKFlipFlopID));
        properties.setProperty("ItemPulseGeneratorID",
Integer.toString(ItemPulseGenID));
        properties.store(fileoutputstream, "MoareAI's Misc Digital
Functions Configurations");
        fileoutputstream.close();
    }
}
}
catch(Throwable throwable) { }

if (!ReduceTexture)
{
    TextureToggleOn = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureToggleOff = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureSROn = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureRSOn = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureRSOff = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureClockOn = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureClockOff = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureHalfAdder = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureJKOn = ModLoader.getUniqueSpriteIndex("/terrain.png");
    TextureJKOff = ModLoader.getUniqueSpriteIndex("/terrain.png");
}
else
{
    if (!mod_LogicalGates.ReduceTexture)
        TextureToggleOn = ModLoader.getUniqueSpriteIndex("/terrain.png");
    else
        TextureToggleOn = mod_LogicalGates.TextureNOTOn;
    TextureToggleOff = TextureToggleOn;
    TextureSROn = TextureToggleOn;
    TextureRSOn = TextureToggleOn;
    TextureRSOff = TextureToggleOn;
    TextureClockOn = TextureToggleOn;
    TextureClockOff = TextureToggleOn;
    TextureHalfAdder = TextureToggleOn;
    TextureJKOn = TextureToggleOn;
    TextureJKOff = TextureToggleOn;
}
}

```

```

    if (!ReduceIcon)
    {
        IconToggle = ModLoader.getUniqueSpriteIndex("/gui/items.png");
        IconRS = ModLoader.getUniqueSpriteIndex("/gui/items.png");
        IconClock = ModLoader.getUniqueSpriteIndex("/gui/items.png");
        IconHalfAdder = ModLoader.getUniqueSpriteIndex("/gui/items.png");
        IconJK = ModLoader.getUniqueSpriteIndex("/gui/items.png");
        IconPulseGen = ModLoader.getUniqueSpriteIndex("/gui/items.png");
    }
    else
    {
        if (!mod_LogicalGates.ReduceIcon)
            IconToggle = ModLoader.getUniqueSpriteIndex("/gui/items.png");
        else
            IconToggle = mod_LogicalGates.IconNOT;
        IconRS = IconToggle;
        IconClock = IconToggle;
        IconHalfAdder = IconToggle;
        IconJK = IconToggle;
        IconPulseGen = IconToggle;
    }

    BlockToggle = new BlockDigitalToggle(BlockToggleID, TextureToggleOn,
TextureToggleOff).setBlockName("Toggle");
    BlockRS = new BlockDigitalRS(BlockRSID, TextureSRon, TextureRSOn,
TextureRSOff).setBlockName("RS");
    BlockHalfAdder = new BlockDigitalHalfAdder(BlockHalfAdderID, TextureSRon,
TextureHalfAdder, TextureRSOff).setBlockName("Half Adder");
    BlockJKFlipFlop = new BlockDigitalJKFlipFlop(BlockJKFlipFlopID, TextureJKOn,
TextureJKOff).setBlockName("JK Flip Flop");
    BlockClock = new BlockDigitalClock(BlockClockID, TextureClockOn,
TextureClockOff).setBlockName("Clock");
    BlockPulseGen = new BlockDigitalPulseGen(BlockPulseGenID, TextureToggleOn,
TextureToggleOff).setBlockName("PulseGen");

    ItemToggle = (new ItemDigitalMisc(ItemToggleID, BlockToggle,
IconToggle)).setItemName("Toggle");
    ItemRS = (new ItemDigitalMisc(ItemRSID, BlockRS, IconRS)).setItemName("RS");
    ItemClock = (new ItemDigitalMisc(ItemClockID, BlockClock,
IconClock)).setItemName("Clock");
    ItemHalfAdder = (new ItemDigitalMisc(ItemHalfAdderID, BlockHalfAdder,
IconHalfAdder)).setItemName("Half Adder");
    ItemJKFlipFlop = (new ItemDigitalMisc(ItemJKFlipFlopID, BlockJKFlipFlop,
IconJK)).setItemName("JK Flip Flop");
    ItemPulseGen = (new ItemDigitalMisc(ItemPulseGenID, BlockPulseGen,
IconPulseGen)).setItemName("PulseGen");

}

//The rendering for the gates
public boolean RenderWorldBlock(RenderBlocks renderblocks, IBlockAccess
iblockaccess, int i, int j, int k, Block block, int l)
{
    if(l == RenderId1)
    {
        Tessellator tessellator = Tessellator.instance;
        int i2 = iblockaccess.getBlockMetadata(i, j, k);
        int i1 = BlockBed.getDirectionFromMetadata(i2);
        float f1 = 1.0F;
        float f4 = f1;
        float f5 = f1;
    }
}

```

```

float f6 = f1;
tessellator.setColorOpaque_F(0,0,0);
int j1 = block.getBlockTexture(iblockaccess, i, j, k, 0);
int k1 = (j1 & 0xf) << 4;
int l1 = j1 & 0xf0;
double d = (float)k1 / 256F;
double d1 = ((double)(k1 + 16) - 0.01D) / 256D;
double d2 = (float)l1 / 256F;
double d3 = ((double)(l1 + 16) - 0.01D) / 256D;
double d40 = d;
double d41 = d1;
double d42 = d2;
double d43 = d3;
double d44 = d;
double d45 = d1;
double d46 = d2;
double d47 = d3;
if(i1 % 2 == 1)
{
    d41 = d;
    d42 = d3;
    d44 = d1;
    d47 = d2;
    d43 = d3;
    d40 = d1;
    d45 = d;
    d46 = d2;
} else
if(i1 % 2 == 0)
{
    d40 = d1;
    d43 = d3;
    d45 = d;
    d46 = d2;
}
double d4 = (double)i + block.minX;
double d5 = (double)i + block.maxX;
double d6 = (double)j + block.minY + 0.125D;
double d7 = (double)k + block.minZ;
double d8 = (double)k + block.maxZ;
tessellator.addVertexWithUV(d4, d6, d8, d44, d46);
tessellator.addVertexWithUV(d4, d6, d7, d40, d42);
tessellator.addVertexWithUV(d5, d6, d7, d41, d43);
tessellator.addVertexWithUV(d5, d6, d8, d45, d47);
float f17 = block.getBlockBrightness(iblockaccess, i, j + 1, k);
tessellator.setColorOpaque_F(f4 * f17, f5 * f17, f6 * f17);
k1 = block.getBlockTexture(iblockaccess, i, j, k, 1);
l1 = (k1 & 0xf) << 4;
d = k1 & 0xf0;
double d9 = (float)l1 / 256F;
double d10 = ((double)(l1 + 16) - 0.01D) / 256D;
double d11 = (float)d / 256F;
double d12 = ((d + 16D) - 0.01D) / 256D;
double d13 = d9;
double d14 = d10;
double d15 = d11;
double d16 = d11;
double d17 = d9;
double d18 = d10;
double d19 = d12;
double d20 = d12;
if(i1 == 0)
{
    d14 = d9;

```

```

        d15 = d12;
        d17 = d10;
        d20 = d11;
    } else
    {
        if(i1 == 2)
        {
            d13 = d10;
            d16 = d12;
            d18 = d9;
            d19 = d11;
        } else
        {
            if(i1 == 3)
            {
                d13 = d10;
                d16 = d12;
                d18 = d9;
                d19 = d11;
                d14 = d9;
                d15 = d12;
                d17 = d10;
                d20 = d11;
            }
        }
        double d21 = (double)i + block.minX;
        double d22 = (double)i + block.maxX;
        double d23 = (double)j + block.maxY;
        double d24 = (double)k + block.minZ;
        double d25 = (double)k + block.maxZ;
        tessellator.addVertexWithUV(d22, d23, d25, d17, d19);
        tessellator.addVertexWithUV(d22, d23, d24, d13, d15);
        tessellator.addVertexWithUV(d21, d23, d24, d14, d16);
        tessellator.addVertexWithUV(d21, d23, d25, d18, d20);
        if(i2 % 2 == 1)
        {
            int j2 = (block.getBlockTexture(iblockaccess, i, j, k, 2) & 0xf) << 4;
            int k2 = block.getBlockTexture(iblockaccess, i, j, k, 2) & 0xf0;
            double d30 = (((double)j2 + block.minX * 16D) / 256D;
            double d31 = (((double)j2 + block.maxX * 16D) - 0.01D) / 256D;
            double d32 = (((double)k2 + block.minY * 16D) / 256D;
            double d33 = (((double)k2 + block.maxY * 16D) - 0.01D) / 256D;
            double d34 = i + block.minX;
            double d35 = i + block.maxX;
            double d36 = j + block.minY;
            double d37 = j + block.maxY;
            double d38 = k + block.minZ;
            tessellator.addVertexWithUV(d34, d37, d38, d31, d32);
            tessellator.addVertexWithUV(d35, d37, d38, d30, d32);
            tessellator.addVertexWithUV(d35, d36, d38, d30, d33);
            tessellator.addVertexWithUV(d34, d36, d38, d31, d33);
        }
        if(i2 % 2 == 1)
        {
            int j2 = (block.getBlockTexture(iblockaccess, i, j, k, 3) & 0xf) << 4;
            int k2 = block.getBlockTexture(iblockaccess, i, j, k, 3) & 0xf0;
            double d30 = (((double)j2 + block.minX * 16D) / 256D;
            double d31 = (((double)j2 + block.maxX * 16D) - 0.01D) / 256D;
            double d32 = (((double)k2 + block.minY * 16D) / 256D;
            double d33 = (((double)k2 + block.maxY * 16D) - 0.01D) / 256D;
            double d34 = i + block.minX;
            double d35 = i + block.maxX;
            double d36 = j + block.minY;
            double d37 = j + block.maxY;
            double d38 = k + block.maxZ;
            tessellator.addVertexWithUV(d34, d37, d38, d30, d32);
            tessellator.addVertexWithUV(d34, d36, d38, d30, d33);
        }
    }
}

```



```

        tessellator.addVertexWithUV(d35, d36, d38, d31, d33);
        tessellator.addVertexWithUV(d35, d37, d38, d31, d32);
    }
    if(i2 % 2 == 0)
    {
        int j2 = (block.getBlockTexture(iblockaccess, i, j, k, 4) & 0xf) << 4;
        int k2 = block.getBlockTexture(iblockaccess, i, j, k, 4) & 0xf0;
        double d30 = (((double)j2 + block.minZ * 16D) / 256D;
        double d31 = (((double)j2 + block.maxZ * 16D) - 0.01D) / 256D;
        double d32 = (((double)k2 + block.minY * 16D) / 256D;
        double d33 = (((double)k2 + block.maxY * 16D) - 0.01D) / 256D;
        double d34 = i + block.minX;
        double d35 = j + block.minY;
        double d36 = j + block.maxY;
        double d37 = k + block.minZ;
        double d38 = k + block.maxZ;
        tessellator.addVertexWithUV(d34, d36, d38, d31, d32);
        tessellator.addVertexWithUV(d34, d36, d37, d30, d32);
        tessellator.addVertexWithUV(d34, d35, d37, d30, d33);
        tessellator.addVertexWithUV(d34, d35, d38, d31, d33);
    }
    if(i2 % 2 == 0)
    {
        int j2 = (block.getBlockTexture(iblockaccess, i, j, k, 5) & 0xf) << 4;
        int k2 = block.getBlockTexture(iblockaccess, i, j, k, 5) & 0xf0;
        double d30 = (((double)j2 + block.minZ * 16D) / 256D;
        double d31 = (((double)j2 + block.maxZ * 16D) - 0.01D) / 256D;
        double d32 = (((double)k2 + block.minY * 16D) / 256D;
        double d33 = (((double)k2 + block.maxY * 16D) - 0.01D) / 256D;
        double d34 = i + block.maxX;
        double d35 = j + block.minY;
        double d36 = j + block.maxY;
        double d37 = k + block.minZ;
        double d38 = k + block.maxZ;
        tessellator.addVertexWithUV(d34, d35, d38, d30, d33);
        tessellator.addVertexWithUV(d34, d35, d37, d31, d33);
        tessellator.addVertexWithUV(d34, d36, d37, d31, d32);
        tessellator.addVertexWithUV(d34, d36, d38, d30, d32);
    }
    return true;
}
return false;
}
}

```