

```

package net.minecraft.src;

import java.io.IOException;

// 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);

        //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);

        //Item names
        ModLoader.AddName(ItemToggle, "Toggle");
        ModLoader.AddName(ItemRS, "RS Latch");
        ModLoader.AddName(ItemClock, "Pulse Clock");
        ModLoader.AddName(ItemHalfAdder, "Half Adder");
        ModLoader.AddName(ItemJKFlipFlop, "JK Flip Flop");

        //Recipe for Toggle
        ModLoader.AddRecipe(new ItemStack(ItemToggle, 6), 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, 3), 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, 6), 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
    });
}

// Tests for game version
public String Version()
{
    return "1.5_01";
}

//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;

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

//Properties
private static Properties properties = new Properties();
public static boolean PropNewID;
public static int ClockTime;
public static int JKTrigger;

public static int BlockToggleID;
public static int BlockRSID;
public static int BlockClockID;
public static int BlockHalfAdderID;
public static int BlockJKFlipFlopID;
public static int ItemToggleID;
public static int ItemRSID;
public static int ItemClockID;
public static int ItemHalfAdderID;
public static int ItemJKFlipFlopID;

//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;

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

//Block and Item data
static
{
    LoadProperties();
    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");
    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");

    BlockToggle = new BlockDigitalToggle(BlockToggleID).setBlockName("Toggle");
    BlockRS = new BlockDigitalRS(BlockRSID).setBlockName("RS");
    BlockHalfAdder = new BlockDigitalHalfAdder(BlockHalfAdderID).setBlockName("Half
Adder");
    BlockJKFlipFlop = new
BlockDigitalJKFlipFlop(BlockJKFlipFlopID).setBlockName("JK Flip Flop");
    BlockClock = new BlockDigitalClock(BlockClockID).setBlockName("Clock");

    ItemToggle = (new ItemReed(ItemToggleID,
BlockToggle)).setIconIndex(IconToggle).setItemName("Toggle");
    ItemRS = (new ItemReed(ItemRSID,
BlockRS)).setIconIndex(IconRS).setItemName("RS");
    ItemClock = (new ItemPulseClock(ItemClockID,
BlockClock)).setIconIndex(IconClock).setItemName("Clock");
    ItemHalfAdder = (new ItemReed(ItemHalfAdderID,
BlockHalfAdder)).setIconIndex(IconHalfAdder).setItemName("Half Adder");
    ItemJKFlipFlop = (new ItemReed(ItemJKFlipFlopID,
BlockJKFlipFlop)).setIconIndex(IconJK).setItemName("JK Flip Flop");
}

```

```

//Settings for handling of the property file
private static void LoadProperties()
{
    InputStream inputstream =
(mod_MiscDigital.class).getClassLoader().getResourceAsStream("MoareAI/MiscDigitalFunctions.properties");
    if(inputstream != null)
    {
        try
        {
            properties.load(inputstream);
            System.out.println("Succesfully loaded Proterties for MoareAI's
Miscellaneous Digital Functions");
            BlockToggleID =
Integer.parseInt(properties.getProperty("BlockToggle"));
            BlockRSID = Integer.parseInt(properties.getProperty("BlockRS"));
            BlockClockID = Integer.parseInt(properties.getProperty("BlockClock"));
            BlockHalfAdderID =
Integer.parseInt(properties.getProperty("BlockHalfAdder"));
            BlockJKFlipFlopID =
Integer.parseInt(properties.getProperty("BlockJKFlipFlop"));
            ItemToggleID = Integer.parseInt(properties.getProperty("ItemToggle"));
            ItemRSID = Integer.parseInt(properties.getProperty("ItemRS"));
            ItemClockID = Integer.parseInt(properties.getProperty("ItemClock"));
            ItemHalfAdderID =
Integer.parseInt(properties.getProperty("ItemHalfAdder"));
            ItemJKFlipFlopID =
Integer.parseInt(properties.getProperty("ItemJKFlipFlop"));
            ClockTime =
Integer.parseInt(properties.getProperty("ClockPulseWidth"));
            JKTrigger =
Integer.parseInt(properties.getProperty("TriggerJKFlipFlop"));
            System.out.println((new StringBuilder()).append("Pulse Clock
pulsewidth: ").append(ClockTime).toString());
            System.out.println((new StringBuilder()).append("JK Flip Flop Trigger:
").append(JKTrigger).toString());
            System.out.println((new StringBuilder()).append("Block: Toggle ID:
").append(BlockToggleID).toString());
            System.out.println((new StringBuilder()).append("Block: RS Latch ID:
").append(BlockRSID).toString());
            System.out.println((new StringBuilder()).append("Block: Pulse Clock ID:
").append(BlockClockID).toString());
            System.out.println((new StringBuilder()).append("Block: Half Adder ID:
").append(BlockHalfAdderID).toString());
            System.out.println((new StringBuilder()).append("Block: JK Flip Flop
ID: ").append(BlockJKFlipFlopID).toString());
            System.out.println((new StringBuilder()).append("Item: Toggle ID:
").append(ItemToggleID).toString());
            System.out.println((new StringBuilder()).append("Item: RS Latch ID:
").append(ItemRSID).toString());
            System.out.println((new StringBuilder()).append("Item: Pulse Clock ID:
").append(ItemClockID).toString());
            System.out.println((new StringBuilder()).append("Item: Half Adder ID:
").append(ItemHalfAdderID).toString());
            System.out.println((new StringBuilder()).append("Item: JK Flip Flop ID:
").append(ItemJKFlipFlopID).toString());
            return;
        }
        catch(IOException ioexception)
        {
            System.out.println("Failed to load Properties for MoareAI's
Miscellaneous Digital Functions");
        }
    }
}

```

```

    }
}

//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;
}
}

```