

```

package net.minecraft.src;

// Created by MoareAI

import java.util.Random;

public class BlockDigitalHalfAdder extends Block
{
    //Class parameters
    protected BlockDigitalHalfAdder(int id)
    {
        super(id, 147, Material.circuits);
        setBlockBounds(0.0F, 0.0F, 0.0F, 1.0F, 0.125F, 1.0F);
        setLightValue(0.625F);
        setHardness(0.0F);
    }

    //Use render types
    public boolean renderAsNormalBlock()
    {
        return false;
    }

    //Block texture
    public int getBlockTextureFromSideAndMetadata(int side, int meta)
    {
        if(meta > 7)
            return blockIndexInTexture = mod_MiscDigital.TextureSRon;
        else
            if(meta > 3)
                return blockIndexInTexture = mod_MiscDigital.TextureHalfAdder;
            return blockIndexInTexture = mod_MiscDigital.TextureRSOff;
    }

    //Block type (preset)
    public int getRenderType()
    {
        return mod_MiscDigital.RenderGate;
    }

    //The function. Sets metadata to +4 when the gate is to be turned on
    public void onNeighborBlockChange(World world, int x, int y, int z, int id)
    {
        if(!canBlockStay(world, x, y, z))
        {
            dropBlockAsItem(world, x, y, z, world.getBlockMetadata(x, y, z));
            world.setBlockWithNotify(x, y, z, 0);
            return;
        }
        int meta = world.getBlockMetadata(x, y, z);
        boolean InputS = func_InputS(world, x, y, z, meta);
        boolean InputR = func_InputR(world, x, y, z, meta);
        if(InputS && InputR)
        {
            if(meta <= 3)
                world.setBlockMetadataWithNotify(x, y, z, meta+8);
            else
                if(meta <= 7)
                    world.setBlockMetadataWithNotify(x, y, z, meta+4);
        }
        else
            if(InputS ^ InputR)
            {
                if(meta <= 3)

```

BlockDigitalHalfAdder.java

```

        world.setBlockMetadataWithNotify(x, y, z, meta+4);
    else
    if(meta > 7)
        world.setBlockMetadataWithNotify(x, y, z, meta-4);
    }
    else
    {
        if(meta > 7)
            world.setBlockMetadataWithNotify(x, y, z, meta-8);
        else
        if(meta > 3)
            world.setBlockMetadataWithNotify(x, y, z, meta-4);
    }
    world.scheduleBlockUpdate(x, y, z, blockID, 0);
}

public boolean isPoweringTo(IBlockAccess iblockaccess, int i, int j, int k, int
side)
{
    int meta = iblockaccess.getBlockMetadata(i, j, k);
    int rotation = meta%4;
    if(meta < 4)
    {
        return false;
    }
    if(((rotation == 1 && meta >= 7) || (rotation == 0 && meta >= 4 && meta < 8))
&& side == 3)
    {
        return true;
    }
    if(((rotation == 2 && meta >= 7) || (rotation == 1 && meta >= 4 && meta < 8))
&& side == 4)
    {
        return true;
    }
    if(((rotation == 3 && meta >= 7) || (rotation == 2 && meta >= 4 && meta < 8))
&& side == 2)
    {
        return true;
    }
    return (((rotation == 0 && meta >= 7) || (rotation == 3 && meta >= 4 && meta <
8)) && side == 5);
}

//Signal in
//Back - Set
private boolean func_InputS(World world, int x, int y, int z, int meta)
{
    int side = meta%4;
    if (side == 0)
        return world.isBlockIndirectlyProvidingPowerTo(x, y, z+1, 3) ||
((world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z+1, 4) ||
world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z+1, 5)) && world.getBlockId(x, y,
z+1) == Block.redstoneWire.blockID);
    if (side == 1)
        return world.isBlockIndirectlyProvidingPowerTo(x-1, y, z, 4) ||
((world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z+1, 3) ||
world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z-1, 2)) && world.getBlockId(x-1, y,
z) == Block.redstoneWire.blockID);
    if (side == 2)
        return world.isBlockIndirectlyProvidingPowerTo(x, y, z-1, 2) ||
((world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z-1, 4) ||
world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z-1, 5)) && world.getBlockId(x, y, z-
1) == Block.redstoneWire.blockID);
}

```

BlockDigitalHalfAdder.java

```

        if (side == 3)
            return world.isBlockIndirectlyProvidingPowerTo(x+1, y, z, 5) ||
((world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z-1, 3) ||
world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z+1, 2)) && world.getBlockId(x+1, y,
z) == Block.redstoneWire.blockID);
        return false;
    }

    //Right - Reset
    private boolean func_InputR(World world, int x, int y, int z, int meta)
    {
        int side = meta%4;
        if (side == 0)
            return world.isBlockIndirectlyProvidingPowerTo(x+1, y, z, 5) ||
((world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z-1, 3) ||
world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z+1, 2)) && world.getBlockId(x+1, y,
z) == Block.redstoneWire.blockID);
        if (side == 1)
            return world.isBlockIndirectlyProvidingPowerTo(x, y, z+1, 3) ||
((world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z+1, 4) ||
world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z+1, 5)) && world.getBlockId(x, y,
z+1) == Block.redstoneWire.blockID);
        if (side == 2)
            return world.isBlockIndirectlyProvidingPowerTo(x-1, y, z, 4) ||
((world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z+1, 3) ||
world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z-1, 2)) && world.getBlockId(x-1, y,
z) == Block.redstoneWire.blockID);
        if (side == 3)
            return world.isBlockIndirectlyProvidingPowerTo(x, y, z-1, 2) ||
((world.isBlockIndirectlyProvidingPowerTo(x-1, y-1, z-1, 4) ||
world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z-1, 5)) && world.getBlockId(x, y, z-
1) == Block.redstoneWire.blockID);
        return false;
    }

    //Rotation when placed
    public void onBlockPlacedBy(World world, int x, int y, int z, EntityLiving
entityliving)
    {
        int meta = ((MathHelper.floor_double((double)((entityliving.rotationYaw * 4F) /
360F) + 0.5D) & 3) + 2) % 4;
        world.setBlockMetadataWithNotify(x, y, z, meta);
        int id = blockID;
        onNeighborBlockChange(world, x, y, z, id);
    }

    //Notifies neighbor blocks when added
    public void onBlockAdded(World world, int x, int y, int z)
    {
        world.notifyBlocksOfNeighborChange(x+1, y, z, blockID);
        world.notifyBlocksOfNeighborChange(x-1, y, z, blockID);
        world.notifyBlocksOfNeighborChange(x, y, z+1, blockID);
        world.notifyBlocksOfNeighborChange(x, y, z-1, blockID);
        world.notifyBlocksOfNeighborChange(x, y-1, z, blockID);
        world.notifyBlocksOfNeighborChange(x, y+1, z, blockID);
    }

    //Dropped
    public int idDropped(int i, Random random)
    {
        return mod_MiscDigital.ItemHalfAdder.shiftedIndex;
    }

```

BlockDigitalHalfAdder.java

```
//Tests if the block is solid
public boolean isOpaqueCube()
{
    return false;
}

//Tests if the block may provide power. Used for drawing Redstone Wires.
public boolean canProvidePower()
{
    return true;
}

//Tests where the block may be placed
public boolean canPlaceBlockAt(World world, int x, int y, int z)
{
    if(!world.isBlockOpaqueCube(x, y - 1, z))
        return false;
    else
        return super.canPlaceBlockAt(world, x, y, z);
}

//Tests if the block may stay in the world
public boolean canBlockStay(World world, int x, int y, int z)
{
    if(!world.isBlockOpaqueCube(x, y - 1, z))
        return false;
    else
        return super.canBlockStay(world, x, y, z);
}
}
```