

BlockDigitalClock.java

```

package net.minecraft.src;

// Created by MoareAI

import java.util.Random;

public class BlockDigitalClock extends BlockContainer
{
    //Class parameters
    protected BlockDigitalClock(int id, int textureon, int textureoff)
    {
        super(id, mod_LogicalGates.TextureGateOff, Material.circuits);
        setBlockBounds(0.0F, 0.0F, 0.0F, 1.0F, 0.125F, 1.0F);
        setLightValue(0.625F);
        setHardness(0.0F);
        if (!mod_MiscDigital.ReduceTexture)
        {
            TextureOn = textureon;
            TextureOff = textureoff;
        }
        else
        {
            TextureOn = mod_LogicalGates.TextureGateOn;
            TextureOff = mod_LogicalGates.TextureGateOff;
        }
        setTickOnLoad(true);
    }

    //Block texture
    public int getBlockTextureFromSideAndMetadata(int side, int meta)
    {
        if(meta > 3)
            return blockIndexInTexture = TextureOn;
        return blockIndexInTexture = TextureOff;
    }

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

    //Updates on/off
    public void updateTick(World world, int x, int y, int z, Random random)
    {
        int meta = world.getBlockMetadata(x, y, z);
        int NewMeta = meta;
        boolean InputB = func_InputB(world, x, y, z, meta);
        boolean WireB = isPowerConnectedB(world, x, y, z, meta);
        TileEntity tileentity = world.getBlockTileEntity(x, y, z);
        if(meta <= 3 && (InputB || !WireB))
        {
            if (meta <= 3)
                NewMeta = meta+4;
        } else
        {
            if (meta > 3)
                NewMeta = meta-4;
        }
        if (meta != NewMeta)
            world.setBlockMetadataWithNotify(x, y, z, NewMeta);
        world.setBlockTileEntity(x, y, z, tileentity);
    }
}

```

BlockDigitalClock.java

```
//Update block
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;
    }
    TileEntityPulseClock tileentitypulseclock =
(TileEntityPulseClock)world.getBlockTileEntity(x, y, z);
    int meta = world.getBlockMetadata(x, y, z);
    boolean InputB = func_InputB(world, x, y, z, meta);
    boolean WireB = isPowerConnectedB (world, x, y, z, meta);
    int ClockTime = tileentitypulseclock.clocktime/2;
    if (InputB || !WireB)
        ClockTime = tileentitypulseclock.clocktime/2;
    else
        ClockTime = 1;
    world.scheduleBlockUpdate(x, y, z, blockID, ClockTime);
}

protected TileEntity getBlockEntity()
{
    return new TileEntityPulseClock();
}

//Signal out
public boolean isPoweringTo(IBlockAccess iblockaccess, int x, int y, int z, int
side)
{
    int meta = iblockaccess.getBlockMetadata(x, y, z);
    if(meta == 4)
        return side == 3;
    if(meta == 5)
        return side == 4;
    if(meta == 6)
        return side == 2;
    if(meta == 7)
        return side == 5;
    else
        return false;
}

//Signal in
//Back
private boolean func_InputB(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.isBlockIndirectlyGettingPowered(x, y-1, z+1)) && 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.isBlockIndirectlyGettingPowered(x-1, y-1, z)) && 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) ||
```

BlockDigitalClock.java

```

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

    //Checks for Redstone Wire connection or powersources
    //Back
    private boolean isPowerConnectedB(IBlockAccess iblockaccess, int x, int y, int z,
int meta)
    {
        int side = meta%4;
        if (side == 0)
            return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x, y, z+1);
        if (side == 1)
            return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x-1, y, z);
        if (side == 2)
            return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x, y, z-1);
        if (side == 3)
            return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x+1, y, z);
        return false;
    }

    public boolean blockActivated(World world, int x, int y, int z, EntityPlayer
entityplayer)
    {
        TileEntityPulseClock tileentitypulseclock =
(TileEntityPulseClock)world.getBlockTileEntity(x, y, z);
        entityplayer.addChatMessage("Pulse Clock");
        ModLoader.OpenGUI(entityplayer, new GuiPulseClock(tileentitypulseclock));
        return true;
    }

    //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)
    {
        super.onBlockAdded(world, x, y, 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)

```

BlockDigitalClock.java

```
{
    return mod_MiscDigital.ItemClock.shiftedIndex;
}

//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);
}

private final int TextureOn;
private final int TextureOff;
}
```