

BlockDigitalMisc.java

```
package net.minecraft.src;

//Created by MoareAI

import java.util.Random;

public class BlockDigitalMisc extends BlockContainer
    implements ICustomTextureBlock
{
    //Class parameters
    public BlockDigitalMisc(int ID)
    {
        super(ID, 113, Material.circuits);
        setBlockBounds(0.0F, 0.0F, 0.0F, 1.0F, 0.125F, 1.0F);
        //setLightValue(0.625F);
        setHardness(0.0F);
        setTickOnLoad(true);
        disableStats();
    }

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

    public int getRenderType()
    {
        return mod_DigitalFunctions.RenderGateID;
    }

    //Location of the texturefile
    public String getTextureFile()
    {
        return mod_DigitalFunctions.BlockTexture;
    }

    public int getBlockTexture(IBlockAccess iblockaccess, int x, int y, int z, int
side)
    {
        int meta = iblockaccess.getBlockMetadata(x, y, z);
        return 16+meta;
    }

    //Time depending functions.
    public void updateTick(World world, int x, int y, int z, Random random)
    {
        int meta = world.getBlockMetadata(x, y, z);
        boolean InputB = Input(world, x, y, z, 1);
        boolean WireB = isPowerConnected(world, x, y, z, 1);
        TileEntity updatetileentity = world.getBlockTileEntity(x, y, z);
        TileEntityDigitalMisc tileentity =
(TileEntityDigitalMisc)world.getBlockTileEntity(x, y, z);

        //Pulse Clock
        if (meta == 3)
        {
            if(!tileentity.OutputA && (InputB || !WireB))
            {
                tileentity.OutputA = true;
            }
            else
            {

```

BlockDigitalMisc.java

```

        tileentity.OutputA = false;
    }
    onNeighborBlockChange(world, x, y, z, blockID);
}

//Pulse Generator
if (meta == 6)
{
    if (tileentity.State2[0])
    {
        tileentity.OutputA = false;
        onNeighborBlockChange(world, x, y, z, blockID);
    }
}

world.setBlockTileEntity(x, y, z, updatetileentity);
world.setBlockMetadataWithNotify(x, y, z, meta);
}

//The functions
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);
    int delay = 0;
    boolean InputA = Input(world, x, y, z, 0);
    boolean InputB = Input(world, x, y, z, 1);
    boolean InputC = Input(world, x, y, z, 2);
    boolean WireA = isPowerConnected(world, x, y, z, 0);
    boolean WireB = isPowerConnected(world, x, y, z, 1);
    boolean WireC = isPowerConnected(world, x, y, z, 2);
    TileEntityDigitalMisc tileentity =
(TileEntityDigitalMisc)world.getBlockTileEntity(x, y, z);

    //Toggle
    if (meta == 1)
    {
        if(InputB)
        {
            if (tileentity.State == 0)
            {
                tileentity.State = 3;
                tileentity.OutputA = true;
                world.setBlockMetadataWithNotify(x, y, z, meta);
            }
            else
            if (tileentity.State == 2)
            {
                tileentity.State = 1;
                tileentity.OutputA = false;
                world.setBlockMetadataWithNotify(x, y, z, meta);
            }
        }
        if(!InputB)
        {
            if (tileentity.State == 3)
            {
                tileentity.State = 2;

```

```

        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
    else
    {
        if (tileentity.State == 1)
        {
            tileentity.State = 0;
            world.setBlockMetadataWithNotify(x, y, z, meta);
        }
    }
}

//RS Latch
if (meta == 2)
{
    if (InputC)
    {
        tileentity.OutputA = false;
        if (!InputB)
            tileentity.OutputB = false;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }

    if (InputB)
    {
        if (!InputC)
            tileentity.OutputA = true;
        tileentity.OutputB = true;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
}

//Pulse Clock
if (meta == 3)
{
    if (InputB || !WireB)
    {
        delay = tileentity.ClockTime/2;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
    else
    {
        delay = 1;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
}

//Half Adder
if (meta == 4)
{
    if (InputB ^ InputC)
    {
        tileentity.OutputA = true;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
    else
    {
        tileentity.OutputA = false;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
    if (InputB && InputC)
    {
        tileentity.OutputB = true;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
}

```

```

    }
    else
    {
        tileentity.OutputB = false;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
}

//JK Flip Flop
if (meta == 5)
{
    boolean Trigger = EdgeTrigger(world, x, y, z, 1);
    if(InputA && Trigger && InputC)
    {
        if (tileentity.State == 0)
        {
            tileentity.State = 3;
            tileentity.OutputA = true;
            world.setBlockMetadataWithNotify(x, y, z, meta);
        }
        else
        if (tileentity.State == 2)
        {
            tileentity.State = 1;
            tileentity.OutputA = false;
            world.setBlockMetadataWithNotify(x, y, z, meta);
        }
    }
    else
    if(InputA && !InputB && InputC)
    {
        if (tileentity.State == 3)
        {
            tileentity.State = 2;
            world.setBlockMetadataWithNotify(x, y, z, meta);
        }
        else
        if (tileentity.State == 1)
        {
            tileentity.State = 0;
            world.setBlockMetadataWithNotify(x, y, z, meta);
        }
    }
    else
    if(InputC && Trigger)
    {
        tileentity.State = 0;
        tileentity.OutputA = false;
    }
    else
    if(InputA && Trigger)
    {
        tileentity.State = 3;
        tileentity.OutputA = true;
    }
}

//Pulse Generator
if (meta == 6)
{
    if (InputB && !tileentity.OutputA && !tileentity.State2[0])
    {
        delay = 0;
        tileentity.OutputA = true;
    }
}

```

BlockDigitalMisc.java

```

        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
    if (tileentity.OutputA);
    {
        delay = tileentity.PulseLength;
        tileentity.State2[0] = true;
    }
    if (!InputB && !tileentity.OutputA)
    {
        tileentity.State2[0] = false;
        delay = 0;
    }
}

//Counter
if (meta == 7)
{
    boolean TriggerA = EdgeTrigger(world, x, y, z, 0);
    boolean TriggerB = EdgeTrigger(world, x, y, z, 1);
    boolean TriggerC = EdgeTrigger(world, x, y, z, 2);

    if (TriggerA && tileentity.State > 0)
        tileentity.State--;

    if (TriggerB && tileentity.State < tileentity.Count)
        tileentity.State++;

    if (TriggerC)
        tileentity.State = 0;

    if (tileentity.State > tileentity.Count)
        tileentity.State = tileentity.Count;

    if (tileentity.State >= tileentity.Count)
    {
        tileentity.OutputA = true;
    }
    else
    {
        tileentity.OutputA = false;
    }
}

tileentity.InputA = InputA;
tileentity.InputB = InputB;
tileentity.InputC = InputC;
tileentity.ConnectionA = WireA;
tileentity.ConnectionB = WireB;
tileentity.ConnectionC = WireC;
world.scheduleBlockUpdate(x, y, z, blockID, delay);
}

//Signal out
public boolean isPoweringTo(IBlockAccess iblockaccess, int x, int y, int z, int
side)
{
    TileEntityDigitalMisc tileentity =
(TileEntityDigitalMisc)iblockaccess.getBlockTileEntity(x, y, z);
    int Rotation = tileentity.Rotation;
    boolean OutputA = tileentity.OutputA;
    boolean OutputB = tileentity.OutputB;
    if(!(OutputA || OutputB))
    {
        return false;
    }
}

```

BlockDigitalMisc.java

```

    }
    if(((Rotation == 1 && OutputB) || (Rotation == 0 && OutputA)) && side == 3)
    {
        return true;
    }
    if(((Rotation == 2 && OutputB) || (Rotation == 1 && OutputA)) && side == 4)
    {
        return true;
    }
    if(((Rotation == 3 && OutputB) || (Rotation == 2 && OutputA)) && side == 2)
    {
        return true;
    }
    return (((Rotation == 0 && OutputB) || (Rotation == 3 && OutputA)) && side ==
5);
}

//Signal in
private boolean Input(World world, int x, int y, int z, int side)
{
    TileEntityDigitalMisc tileentity =
(TileEntityDigitalMisc)world.getBlockTileEntity(x, y, z);
    int rotation = tileentity.Rotation;
    if (rotation == side%4)
        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 (rotation == (side+1)%4)
        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.isBlockIndirectlyGettingPowered(x, y-1, z-1)) && world.getBlockId(x, y, z-1) ==
Block.redstoneWire.blockID);
    if (rotation == (side+2)%4)
        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);
    if (rotation == (side+3)%4)
        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);
    return false;
}

//Rising Edge Trigger
private boolean EdgeTrigger(World world, int x, int y, int z, int side)
{
    TileEntityDigitalMisc tileentity =
(TileEntityDigitalMisc)world.getBlockTileEntity(x, y, z);
    //if (side == 1)
    {
        boolean Input = Input(world, x, y, z, side);
        if (Input && !tileentity.State2[side])
        {
            tileentity.State2[side] = true;
            return true;
        }
    }
    else

```

```

        if (!Input)
        {
            tileentity.State2[side] = false;
        }
    }
    return false;
}

//Checks for Redstone Wire connection or power sources
private boolean isPowerConnected(IBlockAccess iblockaccess, int x, int y, int z,
int side)
{
    TileEntityLogicalGates tileentity =
(TileEntityLogicalGates)iblockaccess.getBlockTileEntity(x, y, z);
    int rotation = tileentity.Rotation;
    if (rotation == side%4)
        return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x-1, y, z);
    if (rotation == (side+1)%4)
        return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x, y, z-1);
    if (rotation == (side+2)%4)
        return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x+1, y, z);
    if (rotation == (side+3)%4)
        return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x, y, z+1);
    return false;
}

//Open GUI
public boolean blockActivated(World world, int x, int y, int z, EntityPlayer
entityplayer)
{
    int Meta = world.getBlockMetadata(x, y, z);
    TileEntityDigitalMisc tileentity =
(TileEntityDigitalMisc)world.getBlockTileEntity(x, y, z);
    if (Meta != 0)
        entityplayer.addChatMessage(ItemDigitalMisc.FunctionName[Meta]);
    else
        entityplayer.addChatMessage("This is not a function. Don't cheat ;)");
    tileentity.Function = Meta;
    if (Meta == 3 || Meta == 6 || Meta == 7)
    {
        ModLoader.OpenGUI(entityplayer, new GuiDigitalMisc(tileentity));
        return true;
    }
    return false;
}

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

//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);
}

```

BlockDigitalMisc.java

```
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);
}

//Notifies neighbor blocks when removed
public void onBlockRemoval(World world, int x, int y, int z)
{
    super.onBlockRemoval(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);
}

//Tileentity
protected TileEntity getBlockEntity()
{
    return new TileEntityDigitalMisc();
}

//Item yielded upon drop
public int idDropped(int i, Random random)
{
    return mod_DigitalFunctions.ItemDigitalMisc.shiftedIndex;
}

//Metadata to damage
protected int damageDropped(int i)
{
    return i;
}

//Tests if the block is transparent. Used for rendering the world behind the block
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);
}
```


BlockDigitalMisc.java

```
}  
}
```