

BlockDigitalMisc.java

```
package MoareAI.Class;

//Created by MoareAI

import java.util.Random;

public class BlockDigitalMisc extends BlockContainer
    implements ITextureProvider
{
    //Class parameters
    public BlockDigitalMisc(int ID)
    {
        super(ID, 16, Material.circuits);
        setBlockBounds(0.0F, 0.0F, 0.0F, 1.0F, 0.125F, 1.0F);
        setHardness(0.0F);
        setTickOnLoad(true);
        disableStats();
    }

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

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

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

    //Block texture
    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, B);
        boolean WireB = isPowerConnected(world, x, y, z, B);
        TileEntity updatetileentity = world.getBlockTileEntity(x, y, z);
        TileEntityDigitalMisc tileentity =
(TileEntityDigitalMisc)world.getBlockTileEntity(x, y, z);

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

```

BlockDigitalMisc.java

```

        tileentity.Output[F] = false;
    }
    onNeighborBlockChange(world, x, y, z, blockID);
}

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

//3-bit Random Generator
if (meta == 8)
{
    boolean TriggerB = EdgeTrigger(world, x, y, z, B);
    boolean RandomBool[] = {random.nextBoolean(), random.nextBoolean(),
random.nextBoolean(), random.nextBoolean()};
    if (TriggerB || ActivateRandom)
    {
        tileentity.Output[G] = RandomBool[G];
        tileentity.Output[F] = RandomBool[F];
        tileentity.Output[I] = RandomBool[I];
        onNeighborBlockChange(world, x, y, z, blockID);
        ActivateRandom = false;
    }
}

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 Input[] = {false, false, false, false};
    Input[A] = Input(world, x, y, z, A);
    Input[B] = Input(world, x, y, z, B);
    Input[C] = Input(world, x, y, z, C);
    boolean WireA = isPowerConnected(world, x, y, z, A);
    boolean WireB = isPowerConnected(world, x, y, z, B);
    boolean WireC = isPowerConnected(world, x, y, z, C);
    TileEntityDigitalMisc tileentity =
(TileEntityDigitalMisc)world.getBlockTileEntity(x, y, z);

    //Toggle
    if (meta == 1)
    {
        if(Input[B])
        {
            if (tileentity.State2 == 0)
            {

```

BlockDigitalMisc.java

```

        tileentity.State2 = 3;
        tileentity.Output[F] = true;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
    else
    {
        if (tileentity.State2 == 2)
        {
            tileentity.State2 = 1;
            tileentity.Output[F] = false;
            world.setBlockMetadataWithNotify(x, y, z, meta);
        }
    }
    if (!Input[B])
    {
        if (tileentity.State2 == 3)
        {
            tileentity.State2 = 2;
            world.setBlockMetadataWithNotify(x, y, z, meta);
        }
        else
        {
            if (tileentity.State2 == 1)
            {
                tileentity.State2 = 0;
                world.setBlockMetadataWithNotify(x, y, z, meta);
            }
        }
    }
}

//RS Latch
if (meta == 2)
{
    byte C1 = C;
    byte G1 = G;
    if (tileentity.State[2])
    {
        C1 = A;
        G1 = I;
    }
    if (Input[C1])
    {
        tileentity.Output[F] = false;
        if (!Input[B])
            tileentity.Output[G1] = false;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }

    if (Input[B])
    {
        if (!Input[C1])
            tileentity.Output[F] = true;
        tileentity.Output[G1] = true;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
}

//Pulse Clock
if (meta == 3)
{
    if (Input[B] || !WireB)
    {
        delay = tileentity.ClockTime/2;
    }
    else

```

```

    {
        delay = 1;
    }
}

//Half Adder
if (meta == 4)
{
    byte C1 = C;
    byte G1 = G;
    if (tileentity.State[2])
    {
        C1 = A;
        G1 = I;
    }

    if (Input[B] ^ Input[C1])
    {
        tileentity.Output[F] = true;
    }
    else
    {
        tileentity.Output[F] = false;
    }
    if (Input[B] && Input[C1])
    {
        tileentity.Output[G1] = true;
    }
    else
    {
        tileentity.Output[G1] = false;
    }
}

//JK Flip Flop
if (meta == 5)
{
    boolean TriggerB = EdgeTrigger(world, x, y, z, B);
    if (Input[A] && TriggerB && Input[C])
    {
        if (tileentity.State2 == 0)
        {
            tileentity.State2 = 3;
            tileentity.Output[F] = true;
            world.setBlockMetadataWithNotify(x, y, z, meta);
        }
        else
        if (tileentity.State2 == 2)
        {
            tileentity.State2 = 1;
            tileentity.Output[F] = false;
            world.setBlockMetadataWithNotify(x, y, z, meta);
        }
    }
    else
    if (Input[A] && !Input[B] && Input[C])
    {
        if (tileentity.State2 == 3)
        {
            tileentity.State2 = 2;
            world.setBlockMetadataWithNotify(x, y, z, meta);
        }
        else
        if (tileentity.State2 == 1)

```

BlockDigitalMisc.java

```

    {
        tileentity.State2 = 0;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
}
else
if(Input[C] && TriggerB)
{
    tileentity.State2 = 0;
    tileentity.Output[F] = false;
}
else
if(Input[A] && TriggerB)
{
    tileentity.State2 = 3;
    tileentity.Output[F] = true;
}
}

//Pulse Generator
if (meta == 6)
{
    if (Input[B] && !tileentity.Output[F] && !tileentity.State[0])
    {
        delay = 0;
        tileentity.Output[F] = true;
        world.setBlockMetadataWithNotify(x, y, z, meta);
    }
    if (tileentity.Output[F]);
    {
        delay = tileentity.PulseLength;
        tileentity.State[0] = true;
    }
    if (!Input[B] && !tileentity.Output[F])
    {
        tileentity.State[0] = false;
        delay = 0;
    }
}

//Counter
if (meta == 7)
{
    boolean TriggerA = EdgeTrigger(world, x, y, z, A);
    boolean TriggerB = EdgeTrigger(world, x, y, z, B);
    boolean TriggerC = EdgeTrigger(world, x, y, z, C);

    if (TriggerA && tileentity.State2 > 0)
        tileentity.State2--;

    if (TriggerB && tileentity.State2 < tileentity.Count)
        tileentity.State2++;

    if (TriggerC)
        tileentity.State2 = 0;

    if (tileentity.State2 > tileentity.Count)
        tileentity.State2 = tileentity.Count;

    if (tileentity.State2 >= tileentity.Count)
    {
        tileentity.Output[F] = true;
    }
    else

```

```

        {
            tileentity.Output[F] = false;
        }
    }

    tileentity.Type = meta;
    tileentity.Input[A] = Input[A];
    tileentity.Input[B] = Input[B];
    tileentity.Input[C] = Input[C];
    tileentity.Connection[A] = WireA;
    tileentity.Connection[B] = WireB;
    tileentity.Connection[C] = 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 Output[] = {false, false, false, false};
    Output[F] = tileentity.Output[F];
    Output[G] = tileentity.Output[G];
    Output[I] = tileentity.Output[I];
    if(!(Output[F] || Output[G] || Output[I]))
        return false;
    if(side == 3)
        return ((Rotation == 0 && Output[F]) || (Rotation == 1 && Output[G]) ||
(Rotation == 3 && Output[I]));
    if(side == 4)
        return ((Rotation == 1 && Output[F]) || (Rotation == 2 && Output[G]) ||
(Rotation == 0 && Output[I]));
    if(side == 2)
        return ((Rotation == 2 && Output[F]) || (Rotation == 3 && Output[G]) ||
(Rotation == 1 && Output[I]));
    if (side == 5)
        return ((Rotation == 3 && Output[F]) || (Rotation == 0 && Output[G]) ||
(Rotation == 2 && Output[I]));
    return false;
}

//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+3)%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+0)%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+1)%4)
        return world.isBlockIndirectlyProvidingPowerTo(x+1, y, z, 5) ||
((world.isBlockIndirectlyProvidingPowerTo(x+1, y-1, z-1, 3) ||

```

BlockDigitalMisc.java

```

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+2)%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);
    {
        boolean Input = Input(world, x, y, z, side);
        if (Input && !tileentity.State[side])
        {
            tileentity.State[side] = true;
            return true;
        }
        else
        if (!Input)
        {
            tileentity.State[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)
{
    TileEntityDigitalMisc tileentity =
(TileEntityDigitalMisc)iblockaccess.getBlockTileEntity(x, y, z);
    int rotation = tileentity.Rotation;
    if (rotation == (side+3)%4)
    {
        //Redstone Repeater fix
        if (iblockaccess.getBlockId(x-1, y, z) ==
Block.redstoneRepeaterActive.blockID || iblockaccess.getBlockId(x-1, y, z) ==
Block.redstoneRepeaterIdle.blockID)
        {
            if (iblockaccess.getBlockMetadata(x-1, y, z) == 1)
                return true;
            else
                return false;
        }
        //Logical Gates fix
        if (ModLoader.isModLoaded("mod_LogicalGates"))
        {
            if (iblockaccess.getBlockId(x-1, y, z) ==
mod_LogicalGates.LogicalGatesBlockID)
            {
                TileEntityLogicalGates tileentity2 =
(TileEntityLogicalGates)iblockaccess.getBlockTileEntity(x-1, y, z);
                if (tileentity2.Rotation != 1 && tileentity2.Type != 9)
                    return false;
            }
        }
    }
}

```

BlockDigitalMisc.java

```

//Digital functions fix
if (iblockaccess.getBlockId(x-1, y, z) ==
mod_DigitalFunctions.DigitalFunctionsBlockID)
{
    TileEntityDigitalMisc tileentity3 =
(TileEntityDigitalMisc)iblockaccess.getBlockTileEntity(x-1, y, z);
    if ((tileentity3.Type == 2 || tileentity3.Type == 4) &&
tileentity3.Rotation == 2)
        return true;
    else
        if (tileentity3.Type == 8 && tileentity3.Rotation != 3)
            return true;
        else
            if (tileentity3.Rotation != 1)
                return false;
    }
    return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x-1, y, z, 0);
}
if (rotation == (side+0)%4)
{
    //Redstone Repeater fix
    if (iblockaccess.getBlockId(x, y, z-1) ==
Block.redstoneRepeaterActive.blockID || iblockaccess.getBlockId(x, y, z-1) ==
Block.redstoneRepeaterIdle.blockID)
    {
        if (iblockaccess.getBlockMetadata(x, y, z-1) == 2)
            return true;
        else
            return false;
    }
    //Logical Gates fix
    if (ModLoader.isModLoaded("mod_LogicalGates"))
    {
        if (iblockaccess.getBlockId(x, y, z-1) ==
mod_LogicalGates.LogicalGatesBlockID)
        {
            TileEntityLogicalGates tileentity2 =
(TileEntityLogicalGates)iblockaccess.getBlockTileEntity(x, y, z-1);
            if (tileentity2.Rotation != 2 && tileentity2.Type != 9)
                return false;
        }
    }
    //Digital functions fix
    if (iblockaccess.getBlockId(x, y, z-1) ==
mod_DigitalFunctions.DigitalFunctionsBlockID)
    {
        TileEntityDigitalMisc tileentity3 =
(TileEntityDigitalMisc)iblockaccess.getBlockTileEntity(x, y, z-1);
        if ((tileentity3.Type == 2 || tileentity3.Type == 4) &&
tileentity3.Rotation == 3)
            return true;
        else
            if (tileentity3.Type == 8 && tileentity3.Rotation != 0)
                return true;
            else
                if (tileentity3.Rotation != 2)
                    return false;
    }
    return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x, y, z-1, 0);
}
if (rotation == (side+1)%4)
{
    //Redstone Repeater fix
    if (iblockaccess.getBlockId(x+1, y, z) ==

```



```

Block.redstoneRepeaterActive.blockID || iblockaccess.getBlockId(x+1, y, z) ==
Block.redstoneRepeaterIdle.blockID)
{
    if (iblockaccess.getBlockMetadata(x+1, y, z) == 3)
        return true;
    else
        return false;
}
//Logical Gates fix
if (ModLoader.isModLoaded("mod_LogicalGates"))
{
    if (iblockaccess.getBlockId(x+1, y, z) ==
mod_LogicalGates.LogicalGatesBlockID)
    {
        TileEntityLogicalGates tileentity2 =
(TileEntityLogicalGates)iblockaccess.getBlockTileEntity(x+1, y, z);
        if (tileentity2.Rotation != 3 && tileentity2.Type != 9)
            return false;
    }
}
//Digital functions fix
if (iblockaccess.getBlockId(x+1, y, z) ==
mod_DigitalFunctions.DigitalFunctionsBlockID)
{
    TileEntityDigitalMisc tileentity3 =
(TileEntityDigitalMisc)iblockaccess.getBlockTileEntity(x+1, y, z);
    if ((tileentity3.Type == 2 || tileentity3.Type == 4) &&
tileentity3.Rotation == 0)
        return true;
    else
        if (tileentity3.Type == 8 && tileentity3.Rotation != 1)
            return true;
        else
            if (tileentity3.Rotation != 3)
                return false;
}
return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x+1, y, z, 0);
}
if (rotation == (side+2)%4)
{
    //Redstone Repeater fix
    if (iblockaccess.getBlockId(x, y, z+1) ==
Block.redstoneRepeaterActive.blockID || iblockaccess.getBlockId(x, y, z+1) ==
Block.redstoneRepeaterIdle.blockID)
    {
        if (iblockaccess.getBlockMetadata(x, y, z+1) == 0)
            return true;
        else
            return false;
    }
}
//Logical Gates fix
if (ModLoader.isModLoaded("mod_LogicalGates"))
{
    if (iblockaccess.getBlockId(x, y, z+1) ==
mod_LogicalGates.LogicalGatesBlockID)
    {
        TileEntityLogicalGates tileentity2 =
(TileEntityLogicalGates)iblockaccess.getBlockTileEntity(x, y, z+1);
        if (tileentity2.Rotation != 0 && tileentity2.Type != 9)
            return false;
    }
}
//Digital functions fix
if (iblockaccess.getBlockId(x, y, z+1) ==

```

```

mod_DigitalFunctions.DigitalFunctionsBlockID)
{
    TileEntityDigitalMisc tileentity3 =
(TileEntityDigitalMisc)iblockaccess.getBlockTileEntity(x, y, z+1);
    if ((tileentity3.Type == 2 || tileentity3.Type == 4) &&
tileentity3.Rotation == 1)
        return true;
    else
        if (tileentity3.Type == 8 && tileentity3.Rotation != 2)
            return true;
        else
            if (tileentity3.Rotation != 0)
                return false;
    }
    return BlockRedstoneWire.isPowerProviderOrWire(iblockaccess, x, y, z+1, 0);
}
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.Type = Meta;
    if ((Meta >= 2 && Meta <= 4) || Meta == 6 || Meta == 7)
    {
        if (world.multiplayerWorld)
        {
            entityplayer.addChatMessage("GUIs currently don't work in Multiplayer.
Default values only.");
            return false;
        }
        ModLoader.OpenGUI(entityplayer, new GuiDigitalMisc(tileentity));
        return false;
    }
    if (Meta == 8)
    {
        if (!world.multiplayerWorld)
        {
            this.ActivateRandom = true;
            onNeighborBlockChange(world, x, y, z, blockID);
        }
        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);
    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);
}

```

BlockDigitalMisc.java

```
//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);
}

public static byte A = 1;
public static byte B = 2;
public static byte C = 3;
public static byte D = 0;
public static byte F = D;
public static byte G = A;
public static byte H = B;
public static byte I = C;
public boolean ActivateRandom;
}
```