雷达数据格式说明

1. 二进制数据组织格式

序号	名称	类型	数值示例	长度	位置
1	固定头	char[4]	1TZX	4	0
2	文件总长度,total	unsigned long long		8	4
3	左上角longitude,left	float		4	12
4	左上角latitude,top	float		4	16
5	右下角longitude,right	float		4	20
6	右下角latitude,bottom	float		4	24
7	经度方向步长,xstep	float		4	28
8	纬度方向步长,ystep	float		4	32
9	预留字段	char[64]		64	36
10	行数,rows	unsigned long long		8	100
11	列数,cols	unsigned long long		8	108
12	压缩后数据长度,cps_len	unsigned long long		8	116
13	LZ4压缩后网格数据块	unsigned char		与12(压缩后数据大小)一致	

2. 补充说明

- 固定头始终为`_TZX`;
- 文件总长度指的指所有数据长度,可以查看文件属性中的大小,与文件大小一致;
- 四个角点坐标为经纬度,经度数值范围74.0-135.0之间, 纬度数值范围3.0-54.0之间;
- 存在关系:`right = left + cols * xstep`;
- 存在关系`top = bottom + rows * ystep`;
- 使用LZ4解压后的数据长度应该为:`rows*cols*sizeof(float)`,解压后需要与这个数值比较,做验证;
- 解压后的网格点为float数组,数组从左上角开始,先由西向东, 在由北到南

3. python脚本示例:

```python
import struct
import lz4.block  # 需要安装 lz4 库: pip install lz4

def read_binary_file(file_path):
    with open(file_path, 'rb') as f:
        # 1. 读取固定头 (4 字节)
        fixed_header = f.read(4).decode('utf-8')
        if fixed_header != '1TZX':
            raise ValueError("Invalid file format. Fixed header is not '1TZX'.")

        # 2. 读取文件总长度 (8 字节)
        total_length = struct.unpack('<Q', f.read(8))[0]

        # 3. 读取左上角经度 (4 字节)
        left = struct.unpack('<f', f.read(4))[0]

        # 4. 读取左上角纬度 (4 字节)
        top = struct.unpack('<f', f.read(4))[0]

        # 5. 读取右下角经度 (4 字节)
        right = struct.unpack('<f', f.read(4))[0]

        # 6. 读取右下角纬度 (4 字节)
        bottom = struct.unpack('<f', f.read(4))[0]

        # 7. 读取经度方向步长 (4 字节)
```

```
        xstep = struct.unpack('<f', f.read(4))[0]

        # 8. 读取纬度方向步长 (4 字节)
        ystep = struct.unpack('<f', f.read(4))[0]

        # 9. 读取预留字段 (64 字节)
        reserved = f.read(64)

        # 10. 读取行数 (8 字节)
        rows = struct.unpack('<Q', f.read(8))[0]

        # 11. 读取列数 (8 字节)
        cols = struct.unpack('<Q', f.read(8))[0]

        # 12. 读取压缩后数据长度 (8 字节)
        cps_len = struct.unpack('<Q', f.read(8))[0]

        # 13. 读取 LZ4 压缩后的网格数据块
        compressed_data = f.read(cps_len)

        # 验证文件总长度
        if f.tell() != total_length:
            raise ValueError(f"File length mismatch. Expected {total_length}, but got {f.tell()}.")
        expected_decompressed_size = rows * cols * 4  # float 占 4 字节
        # 解压 LZ4 数据
        decompressed_data = lz4.block.decompress(compressed_data,
uncompressed_size=expected_decompressed_size)

        # 验证解压后的数据长度

        if len(decompressed_data) != expected_decompressed_size:
            raise ValueError(f"Decompressed data size mismatch. Expected {expected_decompressed_size},
but got {len(decompressed_data)}.")

        # 将解压后的数据转换为浮点数组
        grid_data = struct.unpack(f'<{rows * cols}f', decompressed_data)

    # 打印提取的信息
    print(f"Fixed Header: {fixed_header}")
    print(f"Total Length: {total_length}")
    print(f"Left Longitude: {left}")
    print(f"Top Latitude: {top}")
    print(f"Right Longitude: {right}")
    print(f"Bottom Latitude: {bottom}")
    print(f"Longitude Step: {xstep}")
    print(f"Latitude Step: {ystep}")
    print(f"Rows: {rows}")
    print(f"Columns: {cols}")
    print(f"Compressed Data Length: {cps_len}")
    print(f"Decompressed Grid Data Size: {len(grid_data)}")



# 示例调用
if __name__ == "__main__":
    file_path = "./202503210750.rgrid"  # 替换为实际文件路径
    try:
        data = read_binary_file(file_path)
    except Exception as e:
        print(f"Error: {e}")
```