SREC File Comparator
Practical Step-by-Step Guide

Embedded Systems Engineer

May 5, 2025

Step 1: Understanding the Task

Q: What exactly are we comparing in SREC files?
A: We need to compare:

- **Data records** (S1, S2, S3) - The actual firmware content
- **Address ranges** - Where data is loaded in memory
- **Checksums** (Optional) - Data integrity verification

Step 2: Setting Up the Project

Q: What Qt modules do we need?
A: Essential Qt includes:

```
#include <QFile>
#include <QTextStream>
#include <QHash>
#include <QByteArray>
#include <QtDebug>  // For debugging output
```

Step 3: Basic File Reading

Q: How to read SREC files line by line?
A: Standard Qt file reading pattern:

```
QFile file("firmware.srec");
if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
    qWarning() << "Failed to open file";
    return;
}
```

```
QTextStream in(&file);
while (!in.atEnd()) {
    QString line = in.readLine().trimmed();
    // Process each line here
}
```

Step 4: Parsing SREC Records

Q: How to extract data from each SREC line?
A: Break down each record type:

```
if (line.startsWith("S0")) {
    // Header record - often contains metadata
}
else if (line.startsWith("S1") || line.startsWith("S2") || line.startsWith("S3")
    // Data record - what we really care about
    char recordType = line[1].toLatin1();
    int dataLength = line.mid(2,2).toInt(nullptr,16);
    // Continue parsing...
}
```

Step 5: Handling Different Address Sizes

Q: How to manage 16/24/32-bit addresses?
A: Normalize to 32-bit for consistent comparison:

```
quint32 address = 0;
switch(recordType) {
    case '1': // S1 - 16-bit address
        address = line.mid(4,4).toUInt(nullptr,16);
        break;
    case '2': // S2 - 24-bit address
        address = line.mid(4,6).toUInt(nullptr,16);
        break;
    case '3': // S3 - 32-bit address
        address = line.mid(4,8).toUInt(nullptr,16);
        break;
}
```

Step 6: Storing Data for Comparison

Q: What's the best data structure for comparison?
A: Use QHash for O(1) lookups:

```
QHash<quint32, QByteArray> srecData; // address -> data

// Store parsed data
srecData.insert(address, QByteArray::fromHex(line.mid(...)));
```

Step 7: Implementing the Comparison

Q: How to find differences between two files?
A: Compare both directions:

```
void compareFiles(const QHash<quint32,QByteArray>& file1,
                  const QHash<quint32,QByteArray>& file2) {
    // Check for data in file1 not in file2
    for (auto it = file1.begin(); it != file1.end(); ++it) {
        if (!file2.contains(it.key())) {
            qDebug() << "Missing address in file2:" << Qt::hex << it.key();
        }
        else if (file2[it.key()] != it.value()) {
            qDebug() << "Data differs at" << Qt::hex << it.key();
        }
    }

    // Check for data in file2 not in file1 (reverse check)
    // Similar implementation...
}
```

Step 8: Advanced Features

Q: How to add checksum verification?
A: Implement checksum calculation:

```
bool validateChecksum(const QString& line) {
    uint8_t sum = 0;
    // Sum all bytes after 'Sx' except checksum
    for (int i = 2; i < line.length()-2; i += 2) {
        sum += line.mid(i,2).toUShort(nullptr,16);
    }
    // Compare with stored checksum
    return (0xFF - (sum & 0xFF)) == line.right(2).toUShort(nullptr,16);
}
```

Step 9: Generating Useful Output

Q: What output format is most helpful?
A: Multiple options:

- **Console output** for quick debugging
- **CSV/JSON** for tool integration
- **HTML report** with color-coded differences

```
// Example JSON output
void generateJsonReport(const QString& filename) {
    QJsonArray differences;
    // Add difference entries...

    QJsonDocument doc(differences);
    QFile jsonFile(filename);
    jsonFile.open(QIODevice::WriteOnly);
    jsonFile.write(doc.toJson());
}
```

Step 10: Optimization Tips

Q: How to handle very large SREC files?
A: Several strategies:

- Process files line-by-line (already implemented)
- Use memory mapping for huge files
- Implement parallel parsing with QtConcurrent
- Add progress reporting