DQR-222 BLE Dynamic QR Code Display

Programmer's Operation Guide

Item Code: BEI-DQR-222-BLE

Overview

The DQR-222 BLE Display enables you to:

- **Send text commands** (show different QR/payment screens, control settings)
- **Transfer JPEG images** (dynamic QR/image display)
- **Transfer MP3 audio files** (play audio on the device)

Communication is via **BLE (Bluetooth Low Energy)** using specified UUIDs and command protocols.

1. BLE Connection and UUIDs

Device Name: `Bonrix-DQR-222`

Service UUIDs:

- **Write UUID:** `87654321-4321-4321-4321-cba987654321` (send commands/data here)
- **Notify UUID:** `98765432-1234-1234-1234-123456789abc` (receive notifications/events here)

Use a BLE library (like Bleak for Python) to:

- Scan for the device
- Connect to the device
- Start notifications
- Send commands/data to the write UUID

2. Text Commands (Operation Commands)

These are **string-based commands** (usually with parameters, separated by `**`, and ending with `\n`). You send these to the Write UUID. The device responds over the Notify UUID.

A. UI & QR Code Display Commands

Command Key	Command Format	Purpose
1	`WelcomeScreen**<UPI_ID>\n`	Show Welcome screen with UPI
2	`DisplayFailQRCodeScreen**<MOBILE>**<ORDERID>**<DATE>\n`	Show Failure QR screen
3	`DisplaySuccessQRCodeScreen**<MOBILE>**<ORDERID>**<DATE>\n`	Show Success QR screen
4	`DisplayCancelQRCodeScreen**<MOBILE>**<ORDERID>**<DATE>\n`	Show Cancelled QR screen
5	`DisplayQRCodeScreen**<QR_PAY_URL>**<AMOUNT>**<UPI_ID>\n`	Show Payment QR screen

Example for QR Pay Screen:

```
cmd = "DisplayQRCodeScreen**upi://pay?pa=xxxx@upi&pn=Bonrix&cu=INR&am=10&pn=Bonrix%20Software**10**7418529631@icici\n" await client.write_gatt_char(WRITE_UUID, cmd.encode(), response=True)
```

B. File Management and Playback

Command	Usage/Example	Purpose
`play**<MP3_FILENAME>\n`	`play**audio1.mp3\n`	Play specified MP3
`delete**mp3**<MP3_FILENAME>\n`	`delete**mp3**audio1.mp3\n`	Delete MP3 file
`delete**images**<JPEG_FILENAME>\n`	`delete**images**image1.jpg\n`	Delete JPEG file
`fileinfomp3\n`	`fileinfomp3\n`	Get MP3 files list
`fileinfo\n`	`fileinfo\n`	Get JPEG files list

> **Note:** Device responds with file info as JSON. End markers like `end_of_fileinfo` or `end_of_fileinfomp3` signal completion.

C. Volume & Settings

Command	Usage	Purpose
`setvolume**<N>\n`	`setvolume**10\n`	Set volume (1–21)

Command	Usage	Purpose
`getvolume\n`	`getvolume\n`	Query current volume
`+` or `-`	`+` / `-`	Manual volume up/down
`freesize\n`	`freesize\n`	Get free space info
`startrotation\n`	`startrotation\n`	Start rotation (if supported)
`stoprotation\n`	`stoprotation\n`	Stop rotation
`settimer <sec>\n`	`settimer 60\n`	Set timer in seconds

3. JPEG and MP3 File Transfer

There are **two modes** for uploading files: **SPIFFS (Flash Storage)** and **RAM Mode (temporary, not persistent)**.

A. Upload JPEG to SPIFFS

1. **Send command:** `sending <filename> <filesize>\n`
2. **Wait briefly.**
3. **Send file content** in chunks (default: 512 bytes; each BLE packet ≤244 bytes).
4. Device stores JPEG in `/images`.

Sample Python Flow

```
start_cmd = f"sending myimage.jpg 11245\n" await client.write_gatt_char(WRITE_UUID, start_cmd.encode(), response=True) # Then send JPEG file in 244-byte chunks via BLE to the same characteristic.
```

B. Upload MP3 to SPIFFS

1. **Send command:** `sendingaudio <filename> <filesize>\n`
2. **Wait briefly.**
3. **Send MP3 content** in chunks (same as above).
4. Device stores MP3 in `/mp3files`.

C. RAM Mode JPEG Upload

1. **Send command:** `ssf <filename> <filesize>\n`
2. **Send JPEG file data** (device processes as temp/RAM image).

D. RAM Mode MP3 Upload

1. **Send command:** `ssa <filename> <filesize>\n`
2. **Send MP3 file data** (device processes as temp/RAM audio).

> ### Chunked Data Transfer Protocol
>
> - Always send the "start" command first (as above), then stream the file bytes in **small BLE packets** (≤244 bytes per write).
> - For large files, break the data into 512-byte chunks and then further into BLE-sized packets (244 bytes).
> - Use small delay between writes (`await asyncio.sleep(0.01)`).

4. How to Send Commands and Files

A. Sending a Text Command

```
async def send_command(client, command): message_bytes = (command + '\n').encode('utf-8') for i in range(0, len(message_bytes), 20): chunk = message_bytes[i:i+20] await client.write_gatt_char(WRITE_UUID, chunk, response=True) await asyncio.sleep(0.02)
```

B. Uploading a File (Generic Algorithm)

```
async def upload_file(client, start_cmd, file_path): await client.write_gatt_char(WRITE_UUID, start_cmd.encode(), response=True) await asyncio.sleep(0.1) with open(file_path, "rb") as f: while True: chunk = f.read(512) if not chunk: break for i in range(0, len(chunk), 244): packet = chunk[i:i+244] await client.write_gatt_char(WRITE_UUID, packet, response=False) await asyncio.sleep(0)
```

Upload Command Formats:

- For JPEG to SPIFFS: `start_cmd = f"sending {filename} {filesize}\n"`
- For MP3 to SPIFFS: `start_cmd = f"sendingaudio {filename} {filesize}\n"`
- For JPEG RAM: `start_cmd = f"ssf {filename} {filesize}\n"`
- For MP3 RAM: `start_cmd = f"ssa {filename} {filesize}\n"`

5. BLE Notification Handling

- Listen for notifications from the device for responses, file lists, JSON info, or transfer completion markers.
- The device may return JSON-formatted data (e.g., file lists), which should be parsed by the client.

6. Example: Show QR Code and Upload JPEG

```
# Show QR code await send_command(client, "DisplayQRCodeScreen**upi://pay?pa=123@upi&am=10**10**123@upi") # Upload JPEG image filename = "qr_image.jpg" filesize = os.path.getsize(filename) start_cmd = f"sending {filename} {filesize}\n" await upload_file(client, start_cmd, filename)
```

7. Complete Command Reference

Screen/UI Commands

- `WelcomeScreen**<UPI_ID>\n`
- `DisplayFailQRCodeScreen**<MOBILE>**<ORDERID>**<DATE>\n`
- `DisplaySuccessQRCodeScreen**<MOBILE>**<ORDERID>**<DATE>\n`
- `DisplayCancelQRCodeScreen**<MOBILE>**<ORDERID>**<DATE>\n`
- `DisplayQRCodeScreen**<QR_PAY_URL>**<AMOUNT>**<UPI_ID>\n`

File Management Commands

- `fileinfo\n` → JPEG list
- `fileinfomp3\n` → MP3 list
- `delete**images**<JPEG_FILENAME>\n`
- `delete**mp3**<MP3_FILENAME>\n`
- `play**<MP3_FILENAME>\n`

Volume & Settings Commands

- `setvolume**<N>\n`
- `getvolume\n`
- `+` or `-` (volume up/down)
- `freesize\n`
- `startrotation\n`
- `stoprotation\n`
- `settimer <sec>\n`

File Upload Commands

SPIFFS/Flash Storage:

- JPEG: `sending <filename> <filesize>\n` + file data
- MP3: `sendingaudio <filename> <filesize>\n` + file data

RAM Storage:

- JPEG: `ssf <filename> <filesize>\n` + file data
- MP3: `ssa <filename> <filesize>\n` + file data

Programmer Checklist

1. **Scan and Connect:** Use BLE scanner to find `Bonrix-DQR-222` .
2. **Start Notify:** Begin listening on Notify UUID.

3. **Send Commands:** Use proper command format for screen control, file ops, volume, etc.
4. **File Transfer:** For images/audio, always send the command first, then stream file data in small packets.
5. **Handle Responses:** Parse notifications (including JSON) for status or data.
6. **Clean Disconnect:** Always end with disconnect and stop notification cleanly.

8. Summary Operation Table

Operation	Command/Method	Data Flow
Show Welcome	`WelcomeScreen**<UPI_ID>\n`	Command only
Show Fail QR	`DisplayFailQRCodeScreen**MOBILE**ORDERID**DATE`	Command only
Show Success QR	`DisplaySuccessQRCodeScreen**MOBILE**ORDERID**DATE`	Command only
Show Pay QR	`DisplayQRCodeScreen**URL**AMOUNT**UPI_ID\n`	Command only
Play MP3	`play**FILENAME\n`	Command only
Delete MP3	`delete**mp3**FILENAME\n`	Command only
Delete JPEG	`delete**images**FILENAME\n`	Command only
List JPEG files	`fileinfo\n`	Command, then parse JSON from Notify
List MP3 files	`fileinfomp3\n`	Command, then parse JSON from Notify
Set Volume	`setvolume**N\n`	Command only
Get Volume	`getvolume\n`	Command only, response via Notify
Upload JPEG	`sending FILENAME SIZE\n` + file chunks	Command + binary data
Upload MP3	`sendingaudio FILENAME SIZE\n` + file chunks	Command + binary data
RAM JPEG Upload	`ssf FILENAME SIZE\n` + file chunks	Command + binary data
RAM MP3 Upload	`ssa FILENAME SIZE\n` + file chunks	Command + binary data

Programming Tips

- **Always end commands with `\n`.**
- **Always break up large files into BLE-safe packet sizes (≤244 bytes).**
- **Listen for notifications to handle device replies, especially for file uploads and listings.**

- **For UI/QR commands, parameters are**