Prox QR library client (examples)

Current version: 0.2

Parsec QR Library examples for PNR QX29 family of a proximity readers. Author Dmitry Bogdan. For support please contact Support

Manipulating with a library context

Structure `struct qr_context` is designated to communicate with the library. Library user must deal only with a pointer to mentioned structure.

Create context

Methods: *qr_new_ctx()* and *qr_default_ctx()* are responsible for context creation.

Accepted *qr_new_ctx()* argument is a pointer to a private key.

Example create context example with a pre-defined key

```
    qr_AesKey_t qr_zero_key = {
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00};

    struct qr_context *ctx = qr_new_ctx((const qr_AesKey_t *)qr_zero_key);
```

In this above example we create library context with a key *qr_zero_key* filled with zeroes.

qr_new_default_ctx() does not accept any arguments. It just creates context with a Parsec factory secret default key. Here is an example below.

Example create context with a default key

```
struct qr_context *ctx = qr_new_default_ctx();

    if (ctx != NULL)
    {
        // do great things
    }
```

After aquiring ctx pointer, please perform a NULL reference validation.

See a next paragraph for context destroy.

Destroy context

Library context shall be disposed after usage with a *qr_delete_ctx()* method (accepted argument is previously created context).

Destroy context example

```
// create context with the default key
struct qr_context *ctx = qr_new_default_ctx();

if (ctx != NULL) {
    qr_delete_ctx(ctx);
}
```

QR code generation: 4, 7 bytes and 4 bytes with access struct

QR code generation: Generate 4 byte code

Used method: *qr_GenId4()*. Argument list: `struct qr_context *`*context*, `uint32_t` *Identifier*, `uint8_t *` *result_address*.

Please note that encrypted result is ASCII string, so here and in all examples below we will reserve one character for a zero string termination. Standard *printf()* method provided by C libary will print the string for us.

Example 4 byte code QR code creation

```
int example_gen4()
{
    int result = QR_RESULT_ERR;

    uint8_t encrypted_result[QR_DATA_STRUCT_LEN_ASCII + 1];
    encrypted_result[QR_DATA_STRUCT_LEN_ASCII] = 0;

    struct qr_context *ctx = qr_new_default_ctx();

    if (ctx != NULL)
    {
        result = qr_GenId4(ctx, 0x12345678, encrypted_result);
        qr_delete_ctx(ctx);
    }

    if (result == QR_RESULT_OK)
    {
        printf("encrypted with default key, id4: 0x12345678: %s\n", encrypted_result);
    }
    else
```

```
    {
        printf("failed to encrypt id4\n");
    }

    return result;
}
```

Expected result for this gen4 example is a string: A387DFD6A95D6710C5CBAA84DD732CDF6806A07C3D7ADF

https://duckduckgo.com/?q=QR+A387DFD6A95D6710C5CBAA84DD732CDF6806A07C3D7ADF16B7ADF0A
t=ffnt&ia=answer

QR code generation: Generate 7 byte code

Used method: *example_gen7()*. Argument list: `struct qr_context *`*context*, `uint64_t` *Identifier*, `uint8_t *` *result_address*. Synthax is the same as for ID4 with the Identifier size exception: it is *uint64_t* and high byte will be zeroed.

Example 7 byte QR code generation

```
int example_gen4()
{
    int result = QR_RESULT_ERR;

    uint8_t encrypted_result[QR_DATA_STRUCT_LEN_ASCII + 1];
    encrypted_result[QR_DATA_STRUCT_LEN_ASCII] = 0;

    struct qr_context *ctx = qr_new_default_ctx();

    if (ctx != NULL)
    {
        result = qr_GenId7(ctx, 0x123456789ABCDE, encrypted_result);
        qr_delete_ctx(ctx);
    }

    if (result == QR_RESULT_OK)
    {
        printf("encrypted with default key, id7: 0x123456789ABCDE: %s\n", encrypted_result);
    }
    else
    {
        printf("failed to encrypt id7\n");
    }

    return result;
}
```

Expected result for this gen7 example is a string: D09DAB75599CC79FF11854EEE18D32FCCD300F4B9C901D0

Duckduck QR code: https://duckduckgo.com/?q=QR+D09DAB75599CC79FF11854EEE18D32FCCD300F4B9C
t=ffnt&ia=answer

QR code generation: Generate 4 byte code with Access struct

Used method: *example_access()*. Argument list: `struct qr_context *`*context*, `uint32_t` *Identifier*, `qr_UserAccess_t *` *access*, `uint8_t *` *result_address*.

Synthax is the same as for ID4 but with an additional pointer to `qr_UserAccess_t` stucture. This structure is described in qr_usertypes.h header file.

Please fill the `DateTimeFrom` and `DateTimeTo` arrays with a ISO 8601 Date and time fromat. The length of this arrays is limited to 20 bytes and must contain ASCII string with a trailing zero.

Please fill the `TimeStart` and `TimeEnd` array with a ISO 8601 Extended format w/o decimal fraction of the second.

Please do not use multibyte or UTF-8 characters in `DateTimeFrom`, `DateTimeTo` `TimeStart` and `TimeEnd` strings.

If your event is limited to one day, fill `TimeStart` and `TimeEnd` arrays with zeroes since QR code reader will not use this fields. This filds are valid if access time window has more than one day length. The Reader will take into account the first and the last days time values from `DateTimeFrom` and `DateTimeTo` fields with higher priority than those mentioned in `TimeStart` and `TimeEnd`. So if you have two day event, it will be limited from the begin by `DateTimeFrom` till `DateTimeTo` where the second day access window will start from `TimeStart`. If your event has 3 or more working days, the first and the last days will be limited by `DateTimeFrom` and `DateTimeTo` fields but all other days between them will be limited by `TimeStart` and `TimeEnd`.

Area array of Access structure must contain allowed Rooms/Areas list for a visitor. The type of Area is `uint16_t` (values from 0 to 65535), lenght of *Area* array is limited to four elements. Please fill unused elements with zeroes (as in example code).

Example Access QR code generation

```
#include <string.h>

#include "example_main.h"

int example_genaccess()
{
    int result = QR_RESULT_ERR;

    uint8_t encrypted_result[QR_DATA_STRUCT_LEN_ASCII + 1];
```

```
encrypted_result[QR_DATA_STRUCT_LEN_ASCII] = 0;

qr_UserAccess_t access;

// ISO 8601 Date and time from
// "2005-08-09T18:31:42" + trailing zero byte
// Please convert one digit number to two digits:
// 0 seconds, 5 minuts or 8 hours to "00" seconds, "05" minutes and "08" respectively
// or 2nd month or 8th day to "02" month and "08" day
// Please do not use multibyte or UTF-8 characters and terminate string with zero
strncpy(access.DateTimeFrom, "2022-02-14T09:00:42", USER_ACCESS_DATETIME_LEN);
strncpy(access.DateTimeTo, "2022-05-24T00:00:00", USER_ACCESS_DATETIME_LEN);

// Interval during full day
// ISO 8601 Extended format w/o decial fraction of the second
// "18:31:42" + trailing zero byte
// Please do not use single digits values as well (see comment above for DateTime values
// Please do not use multibyte or UTF-8 characters and terminate string with zero
// Fill with all zeroes if you have 1-day interval limited with
// DateTimeFrom and DateTimeTo (you have 1-day event).
//
strncpy(access.TimeStart, "14:33:00", USER_ACCES_TIME_LEN);
strncpy(access.TimeEnd, "15:20:42", USER_ACCES_TIME_LEN);

// List of four allowed Rooms (or Areas) for a visitor
access.Area[0] = 42;
// Please set all to zero if unused
access.Area[1] = 0;
access.Area[2] = 0;
access.Area[3] = 0;

struct qr_context *ctx = qr_new_default_ctx();

if (ctx != NULL)
{
    result = qr_GenAccess(ctx, 0x12345678, &access, encrypted_result);
    qr_delete_ctx(ctx);
}

if (result == QR_RESULT_OK)
{
    printf("encrypted with default key, id4: 0x12345678 with access field Area: 0:%u 1:%
           access.Area[0], access.Area[1], access.Area[2], access.Area[3],
           encrypted_result);
}
else if (result == QR_RESULT_ERR_INVALID_ARGS)
```

```
    {
        printf("failed to encrypt id4 with useraccess, invalid arguments, check your date or
    }
    else
    {
        // unreachable, actually
        printf("failed to encrypt id4 with useraccess with unknown result %d\n", result);
    }

    return result;
}
```

Expected result for this gen_access example is a string: 436E03EF15C8B999A4714C01047364DAE5FC366C30E8C

Duckduck QR code: https://duckduckgo.com/?t=ffab&q=QR+436E03EF15C8B999A4714C01047364DAE5FC36
ia=answer

Light configuration

To perform light sensor and target light LED configuration.

Used method: *qr_GenLightConf()*. Argument list: `struct qr_context *`*context*, `uint8_t` *light_mode*, `uint16_t` *light_threshold*, `uint8_t` *target_mode*,`uint8_t *` *result_address*.

Argument light_mode accepted values are:

Value	description
QR_LIGHT_AUTO_BY_MOTION_SENSOR	by motion sensor
QR_LIGHT_ON	always on
QR_LIGHT_OFF	always off
QR_LIGHT_AUTO_BY_ILLUMINANCE	on when illuminance is lower than threshold

Argument 'light_threshold' accepted values

When `light_mode` is set to `QR_LIGHT_AUTO_BY_ILLUMINANCE`, light_threshold arguments can be set between `QR_LIGHT_THERSHOLD_LUX_LOW` (approx 100 lux) and `QR_LIGHT_THERSHOLD_LUX_HIGH` (approx 500 lux).

Argument target_mode accepted values

Value	description
QR_TARGET_AUTO	by motion sensor
QR_TARGET_ON	always on
QR_TARGET_OFF	always off

Light configuration example

```
int example_genlight() {
    int result = QR_RESULT_ERR;

    uint8_t encrypted_result[QR_DATA_STRUCT_LEN_ASCII + 1];
    encrypted_result[QR_DATA_STRUCT_LEN_ASCII] = 0;

    struct qr_context *ctx = qr_new_default_ctx();

    if (ctx != NULL)
    {
        result = qr_GenLightConf(ctx, QR_LIGHT_AUTO_BY_ILLUMINANCE, 300, QR_TARGET_AUTO, enc
        qr_delete_ctx(ctx);
    }

    if (result == QR_RESULT_OK)
    {
        printf("encrypted light config: %s\n", encrypted_result);
    }
    else
    {
        printf("failed to encrypt light config\n");
    }

    return result;
}
```

Expected result for this set light example is a string: 1AE2BA9B78EB77D2DBA68693B067AE9113C7CA9CD361

Duckduck QR code: https://duckduckgo.com/?q=QR+1AE2BA9B78EB77D2DBA68693B067AE9113C7CA9CD
t=ffnt&ia=answer

QR code generation Set key to a new one

When one wish to change key to a new ona (to perform a default key replacement, *qr_GenSetAesKey()* method shall be used. Library context shall be initialized with a Proximity reader key (context with a default key is used here below).

Set key to a new one example

Here is the example of resetting default key to a key filled with a hexademical zeroes.

```
int example_gensetkey()
{
    int result = QR_RESULT_ERR;
```

```
    qr_AesKey_t qr_zero_key = {
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00};

    uint8_t encrypted_result[QR_DATA_STRUCT_LEN_ASCII + 1];
    encrypted_result[QR_DATA_STRUCT_LEN_ASCII] = 0;

    struct qr_context *ctx = qr_new_default_ctx();

    if (ctx != NULL)
    {
        result = qr_GenSetAesKey(ctx, (const qr_AesKey_t *)qr_zero_key, encrypted_result);
        qr_delete_ctx(ctx);
    }

    if (result == QR_RESULT_OK)
    {
        printf("set default to zero key: %s\n", encrypted_result);
    }
    else
    {
        printf("failed to encrypt zero key\n");
    }

    return result;
}
```

Expected result for this setkey example is a string: 6D9BB6C6AC6C72A7A25AE7A93B75403D39D8BC42CDEF6

Duckduck QR code: https://duckduckgo.com/?q=QR+6D9BB6C6AC6C72A7A25AE7A93B75403D39D8BC42C
t=ffnt&ia=answer

QR code generation Reset key to default

When one wish to change key to default (to perform a key reset), *qr_GenSetDefaultKey()* method shall be used. Library context shall be initialized with a Proximity reader key (filled with zeroes for example).

Reset key from zero to default

Here is the example of resetting zero key to default key:

```
int example_gensetdefaultkey()
{
    int result = QR_RESULT_ERR;
```

```
    qr_AesKey_t qr_zero_key = {
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00};

    uint8_t encrypted_result[QR_DATA_STRUCT_LEN_ASCII + 1];
    encrypted_result[QR_DATA_STRUCT_LEN_ASCII] = 0;

    struct qr_context *ctx = qr_new_ctx((const qr_AesKey_t *)qr_zero_key);

    if (ctx != NULL)
    {
        result = qr_GenSetDefaultAesKey(ctx, encrypted_result);
        qr_delete_ctx(ctx);
    }

    if (result == QR_RESULT_OK)
    {
        printf("reset zero key to default: %s\n", encrypted_result);
    }
    else
    {
        printf("failed to reset zero key to default\n");
    }

    return result;
}
```

Expected result for this setdefkey example is a string:
A60040368F15F3D9912297C4E80E190345B2A555231E5AE8EBC3EEBAB6944F11910AD590B1248816DA6DEB

Duckduck QR code: https://duckduckgo.com/?q=QR+A60040368F15F3D9912297C4E80E190345B2A555231E5A
t=ffnt&ia=answer

QR code generation resources

Duckduck search engine

https://duckduckgo.com

QR-code-generator

https://www.qr-code-generator.com/

The QR code generator

https://www.the-qrcode-generator.com/

Frequently Asked Questions

1. Context reusage

Q. Is it possible to use context (struct qr_context) multiple times?

A. Yes. Configure it once for a new key and use it multiple times till end of the key life. Do not forget to release it with qr_delete_ctx() to upon program termination to prevent memory leaks.