

Gowin goConfig IP(I2C)サンプルコードの 使用方法

TN715-1.0J, 2024-02-02

著作権について(2024)

著作権に関する全ての権利は、**Guangdong Gowin Semiconductor Corporation** に留保されています。

GOWIN及び **Gowin** は、当社により、中国、米国特許商標庁、及びその他の国において登録されています。商標又はサービスマークとして特定されたその他全ての文字やロゴは、それぞれの権利者に帰属しています。何れの団体及び個人も、当社の書面による許可を得ず、本文書の内容の一部もしくは全部を、いかなる視聴覚的、電子的、機械的、複写、録音等の手段によりもしくは形式により、伝搬又は複製をしてはなりません。

免責事項

当社は、**GOWINSEMI Terms and Conditions of Sale**(GOWINSEMI 取引条件)に規定されている内容を除き、(明示的か又は黙示的に拘わらず)いかなる保証もせず、また、知的財産権や材料の使用によりあなたのハードウェア、ソフトウェア、データ、又は財産が被った損害についても責任を負いません。当社は、事前の通知なく、いつでも本文書の内容を変更することができます。本文書を参照する何れの団体及び個人も、最新の文書やエラッタ(不具合情報)については、当社に問い合わせる必要があります。

バージョン履歴

日付	バージョン	説明
2024/02/02	1.0J	初版。

目次

目次	i
図一覧.....	iii
表一覧.....	iv
1 本マニュアルについて	1
1.1 マニュアルの内容	1
1.2 関連ドキュメント	1
1.3 用語、略語.....	1
1.4 テクニカル・サポートとフィードバック	2
2 サンプル・コードの紹介.....	3
2.1 サンプル・コードの概要	3
2.2 ファイル一覧	3
2.2.1 主要ファイルの紹介	4
3 機能の説明	5
3.1 機能の概要.....	5
3.2 コード実行フロー	5
3.2.1 MCU と IP 間の通信が正常かどうかの確認.....	5
3.2.2 バックグラウンドアップグレードの手順	6
4 API の説明.....	12
4.1 よく使用される API のまとめ.....	12
4.2 IP の基本的な命令	12
4.2.1 命令の定義.....	12
4.3 コード読み出し API	17
4.3.1 API の紹介	17
4.3.2 ID Code 読み出しの原理	17
4.3.3 I2C を介してコードを読み出す手順	17
5 サンプル・コードの移植.....	19

5.1 コード・インターフェースの紹介	19
5.1.1 上位層の機能インターフェース	20
5.1.2 中間層インターフェース	20
5.1.3 下位層の I2C 読み出し書き込みインターフェース	20
5.1.4 インターフェースの呼び出し関係	21
5.2 下位層インターフェースの書き換えの説明	21
5.2.1 i2c_init	21
5.2.2 i2c_write_buffer	22
5.2.3 i2c_read_buffer	23
5.2.4 i2c_read_bufferxx	23
5.3 移植のポイント	23
5.3.1 準備	23
5.3.2 コンパイルと検証	24
5.3.3 トラブルシューティング	24
5.3.4 参考タイミング図	25

図一覧

図 3-1 構造	5
図 3-2 MCU と IP 間の通信が正常かどうかを確認するためのフローチャート	6
図 3-3 バックグラウンドアップグレードのフローチャート	7
図 3-4 SRAM コンフィギュレーションのフローチャート	8
図 3-5 オンチップ Flash の消去のフローチャート	9
図 3-6 オンチップ Flash のプログラミングのフローチャート	11
図 4-1 ステータス・レジスタ読み出し命令の例	14
図 4-2 IP リセット命令の例	15
図 4-3 リードバック命令の例	15
図 4-4 FIFO データ読み出し命令の例	16
図 4-5 待機命令の例	16
図 4-6 データを送信して Flash からデータをリードバックしない命令の例	16
図 4-7 Reconfign 命令の例	16
図 4-8 IP バージョン命令	17
図 4-9 パケットの構造	17
図 4-10 I2C を介してコードを読み出すフローチャート	18
図 5-1 インターフェースの呼び出し関係	21
図 5-2 アドレスの確認	25
図 5-3 マクロの定義	25
図 5-4 タイミング図	26
図 5-5 タイミング図	26

表一覧

表 1-1 用語、略語	1
表 2-1 主要ファイル.....	4
表 4-1 よく使用される API のまとめ	12
表 4-2 2C シリーズの命令の定義.....	12
表 4-3 ステータス・レジスタ	14
表 5-1 コード・インターフェースの紹介	19

1 本マニュアルについて

1.1 マニュアルの内容

本マニュアルには、主にサンプル・コードの紹介、機能の説明、APIの説明およびその使用方法が含まれています。

注記：

- このサンプル・コードは、C 言語で作成されており、組み込みプラットフォーム (STM32F429) に基づいた完全なプロジェクト・パッケージです。使用される IDE は keil_V5 です。
- C バージョンの GW1N-1P5/GW1N(R)-2 のコンフィギュレーションに適しています。
- マイクロコントローラ (MCU) が goConfig I2C IP にアクセスする前に、goConfig I2C IP を含むビットストリームファイルを FPGA にロードする必要があります。

1.2 関連ドキュメント

GOWIN セミコンダクターのホームページ www.gowinsemi.com/ja から、以下の関連ドキュメントがダウンロード、参考できます：

- GW1N シリーズ FPGA 製品データシート([DS100](#))
- GW1NR シリーズ FPGA 製品データシート([DS117](#))
- Gowin goConfig I2C IP ユーザーガイド([IPUG795](#))
- Gowin FPGA 製品プログラミング・コンフィギュレーション ユーザーガイド([UG290](#))

1.3 用語、略語

本マニュアルで使用される用語、略語、及びその意味を表 1-1 に示します。

表 1-1 用語、略語

用語、略語	正式名称	意味
ACK	Acknowledge	応答
FPGA	Field Programmable Gate Array	フィールド・プログラマブル・ゲート・アレイ

用語、略語	正式名称	意味
I ² C(I2C)	Inter-Integrated Circuit	2 線式シリアルバス
IP	Intellectual Property	設計資産
NACK	Not Acknowledge	否定応答
SRAM	Static Random Access Memory	スタティック RAM

1.4 テクニカル・サポートとフィードバック

GOWIN セミコンダクターは、包括的な技術サポートをご提供しています。使用に関するご質問、ご意見については、直接弊社までお問い合わせください。

ホームページ : www.gowinsemi.com/ja

E-mail : support@gowinsemi.com

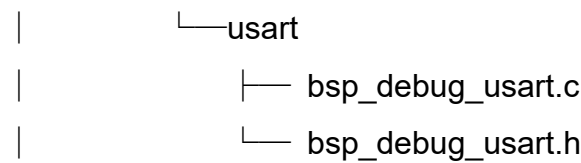
2 サンプル・コードの紹介

2.1 サンプル・コードの概要

このサンプル・コードを移植する場合、カプセル化された上位層 API(`i2c_jtag.c` にある)を直接適用することができます。 下層の I2C データ送信インターフェース(`bsp_i2c.c` にある)は、実際の状況に応じて変更することができます。

2.2 ファイル一覧

```
├──goConfigIP(I2C)サンプル・コードの説明.docx
├──stm32_I2C_PORG_FLASHorSRAM(I2CIP)
│   │   brief introduction.pptx
│   │   readme.txt
│   │   ├──Project
│   │   │   ├──RVMDK (uv5)
│   │   │   │   BH-F429.uvprojx
│   │   │   └──User
│   │   │       │   bitstream.h
│   │   │       │   main.c
│   │   │       │
│   │   │       ├──gw_i2c_xxx
│   │   │       │   i2c_jtag.c
│   │   │       │   i2c_jtag.h
│   │   │       ├──i2c
│   │   │       │   bsp_i2c.c
│   │   │       │   bsp_i2c.h
```



2.2.1 主要ファイルの紹介

表 2-1 主要ファイル

ファイル	説明
User/main.c	main file
User/bitstream.h	the bitstream data in a hpp file
User/User/i2c_jtag	i2c timing file(thru IP)
User/User/i2c/bsp_i2c.c	i2c bsp API

3 機能の説明

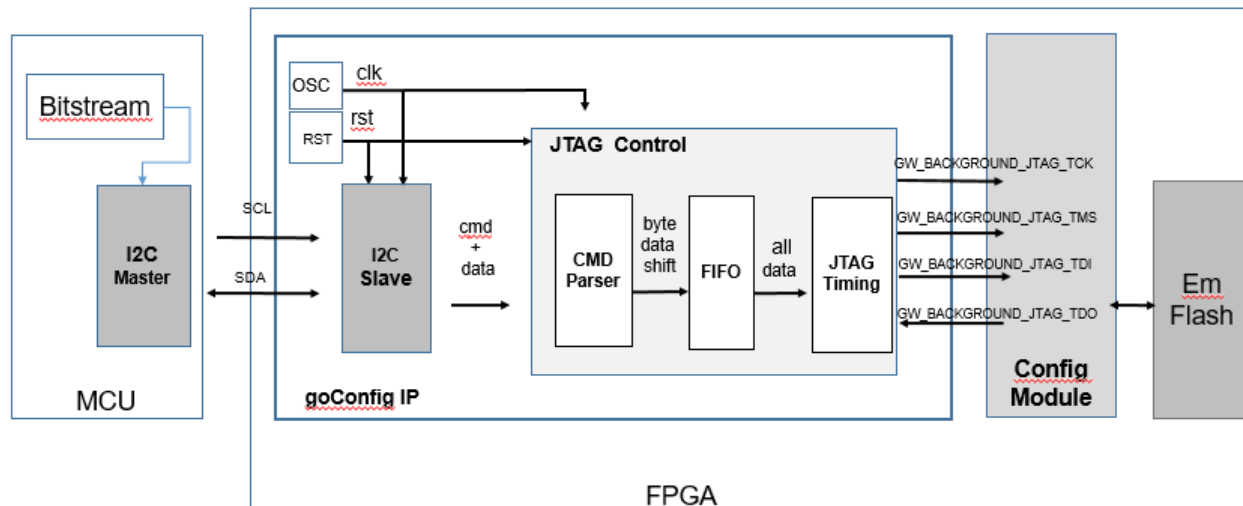
3.1 機能の概要

この例のサンプルコードは MCU で実行され、そして I²C インターフェースを介して goConfigIP を使用してオンチップ Flash の消去とプログラムや、FPGA の ID code, user code, status code の読み出しなどを実現します。

注記：

goConfig IP を使用するには、この IP を含むビットストリームファイルを FPGA にロードする必要があります。

図 3-1 構造



3.2 コード実行フロー

3.2.1 MCU と IP 間の通信が正常かどうかの確認

バックグラウンドアップグレードを開始する前に、まず MCU と IP 間の通信が正常であるかどうかを確認します。

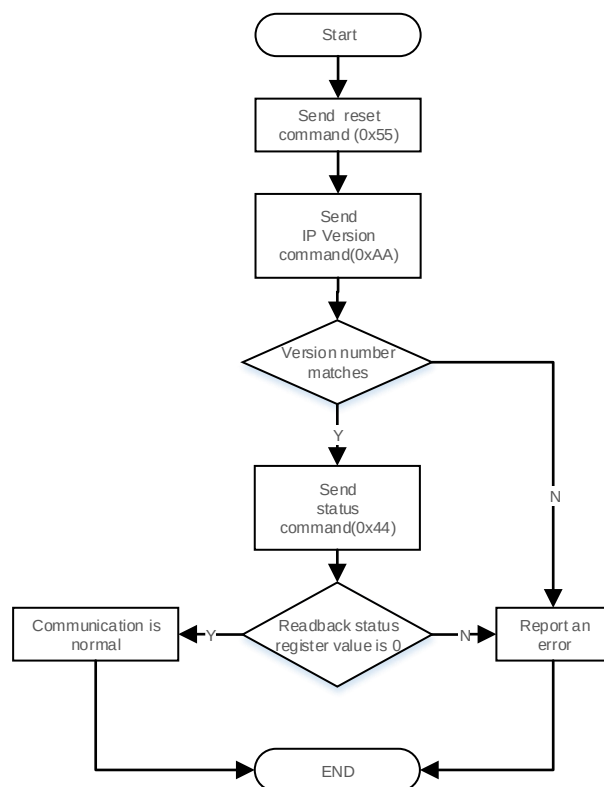
1. Reset 命令 (0x55) を送信して IP を初期状態にします(API : gw_i2c_send_reset)。
2. IP Version 命令(0xAA)を送信して、読み出されたバージョン値 (例えば

Version Data=0x20)が予期したバージョンと一致しているかを確認します (API:gw_i2c_read_version)。

3. Status 命令 (0x44) を送信し、読み出されたステータス値を確認します(API: gw_i2c_read_reg_status)。
4. 操作が完了し、読み出されたデータが期待どおりであれば、IP との通信を開始できます。

MCU と IP 間の通信が正常かどうかを確認するためのフローチャートを図 3-2 に示します。

図 3-2 MCU と IP 間の通信が正常かどうかを確認するためのフローチャート



3.2.2 バックグラウンドアップグレードの手順

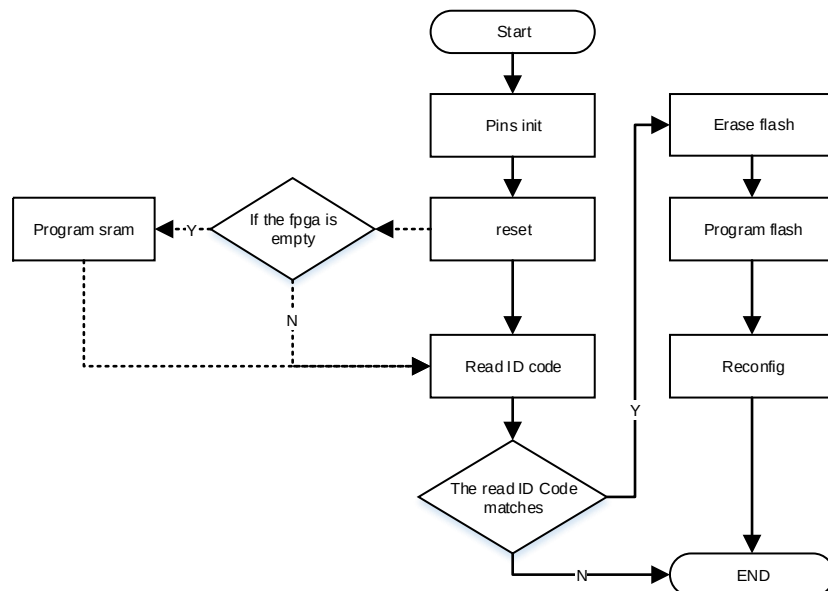
バックグラウンドアップグレードの手順は次のとおりです。

1. MCU プラットフォームのピンを初期化します (I2C、debug_uart)。
2. IP をリセットします (API:gw_i2c_send_reset)。
3. FPGA がブランクチップでない場合は、このステップをスキップしてください。FPGA がブランクチップの場合は、まず SRAM をコンフィギュレーションします(API:i2c_configmode_program_sram)。
4. MCU(I2C)によって読み出された ID code がビットストリームファイルの ID code と一致するかどうかを比較します(API: mach_code)。一致する場合は次のステップへ移行し、一致しない場合はバックグラウンドアップグレードを終了します。
5. Flash を消去します(API: gw_i2c_erase_flash_tsmc)。

6. Flash をプログラムします(API: gw_i2c_prog_flash_tsmc)。
7. 再コンフィギュレーションします(gw_i2c_send_reconfig_n)。

バックグラウンドアップグレードのフローチャートを図 3-3 に示します。

図 3-3 バックグラウンドアップグレードのフローチャート



FPGA Code の読み出し

FPGA code を読み出すには、API: gw_i2c_read_code を使用します。MCU と IP 間の通信が正常であれば、I2C を使用して FPGA code を読み出すことができます。

- ID Code(即ち JEDEC ID Code)は、FPGA デバイスの基本的な ID です。
- user code は、使用する FPGA デバイスのユーザー識別子です。
- status code は、FPGA デバイスがビットストリームファイルのロードを完了した後のステータス識別子です。

例：

- ID code 読み出しの API は、gw_i2c_read_code(0x11)です。
- user code 読み出しの API は、gw_i2c_read_code(0x13)です。
- status code 読み出しの API は、gw_i2c_read_code(0x41)です。

I2C を介したブランクチップの SRAM のコンフィギュレーション

I2C を介してブランクチップの SRAM をコンフィギュレーションするための API : i2c_configmode_program_sram。

注記：

ブランクチップには IP がないので、ここでの SRAM のコンフィギュレーションとは、FPGA 内の専用のハードウェア・コンフィギュレーション・モジュールを通じて SRAM

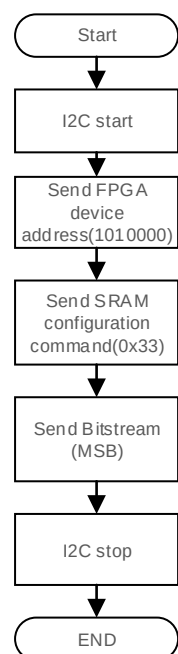
をコンフィギュレーションすることを指します。ハードウェア・コンフィギュレーション・モジュールは、SRAM のコンフィギュレーションのみをサポートします。オンチップ Flash をプログラムするには、goConfig I2C IP を使用する必要があります。

I2C を介して SRAM をコンフィギュレーションする手順は次のとおりです。

1. start 信号を送信します。
2. デバイスのアドレスを送信します。
3. SRAM コンフィギュレーション命令(0x33)を送信します。
4. ビットストリーム (MSB ファースト) を送信します。
5. stop 信号を送信します。

SRAM コンフィギュレーションのフローチャートを図 3-4 に示します。

図 3-4 SRAM コンフィギュレーションのフローチャート



オンチップ Flash の消去

オンチップ Flash を消去するための API : gw_i2c_erase_flash_tsmc。オンチップ Flash を消去する手順は次のとおりです。

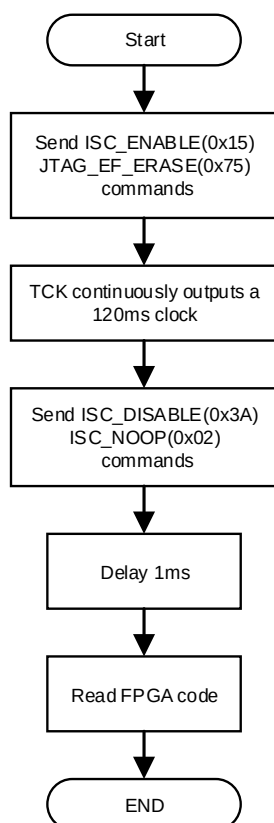
1. ISC_ENABLE(0x15)と JTAG_EF_ERASE(0x75)を送信します。この 2 つの命令は API (write_inst_and_data)によってパッケージ化され、MCU の I2C インターフェース(API: i2c_write_buffer) を介して FPGA に送信されます。
2. TCK でクロックを 120ms 継続的に出力します(API: gw_i2c_send_runttest_xus)。
3. ISC_DISABLE(0x3A)と ISC_NOOP(0x02)を送信します。この 2 つの命

令は API (write_inst)によってパッケージ化され、MCU の I2C インターフェース(API: i2c_write_buffer) を介して FPGA に送信されます。

4. 1ms 遅延します(API: gw_i2c_send_runttest_xus)。
5. FPGA code を読み出します(API : gw_i2c_read_code)。

オンチップ Flash を消去するためのフローチャートを図 3-5 に示します。

図 3-5 オンチップ Flash の消去のフローチャート



オンチップ Flash のプログラミング

オンチップ Flash をプログラムするための API :

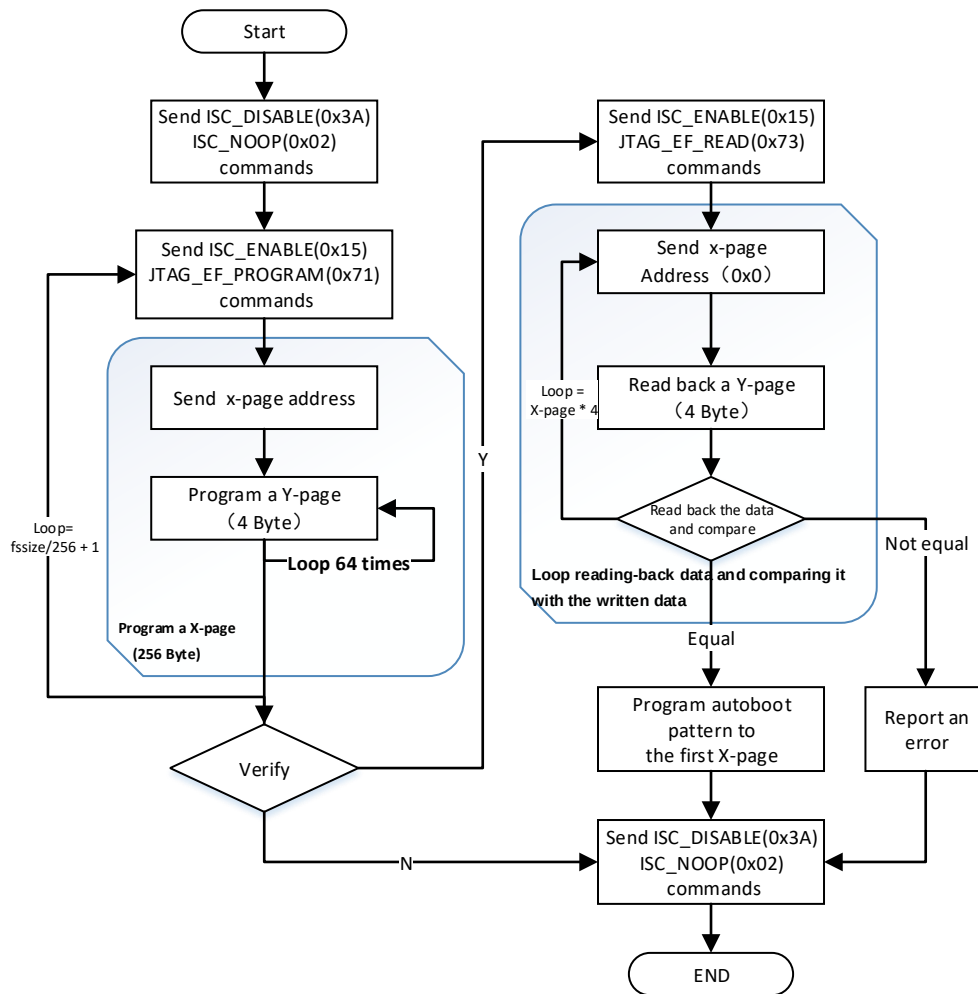
gw_i2c_prog_flash_tsmc。オンチップ Flash をプログラムする手順は次のとおりです。

1. ISC_DISABLE(0x3A)と ISC_NOOP(0x02)を送信します。この 2 つの命令は API (write_inst_and_data)によってパッケージ化され、MCU の I2C インターフェース(API: i2c_write_buffer) を介して FPGA に送信されます。
2. ISC_ENABLE(0x15)と JTAG_EF_PROGRAM(0x71)を送信します。この 2 つの命令は API (write_inst_and_data)によってパッケージ化され、MCU の I2C インターフェース(API: i2c_write_buffer) を介して FPGA に送信されます。
3. X-pgae のアドレスを送信します。このアドレスは、API

- (write_one_x_address)によってパッケージ化され、MCU の I2C インターフェース(API : i2c_write_buffer) を介して FPGA に送信されます。
4. 1 つの Y-page をプログラムします。1 つの X-page には 64 個の Y-page が含まれています。この Y-page のデータは、API (write_one_y_page)によってパッケージ化され、MCU の I2C インターフェース(API : i2c_write_buffer) を介して FPGA に送信されます。
 5. リードバックするには、ISC_ENABLE(0x15)と JTAG_EF_READ(0x73)を送信する必要があります。これら 2 つの命令は API (write_inst_and_data)によってパッケージ化され、MCU の I2C インターフェース(API: i2c_write_buffer) を介して FPGA に送信されます。
 6. X-page のアドレス(0x00)を送信します。このアドレスは、API (write_one_x_address)によってパッケージ化され、MCU の I2C インターフェース (API : i2c_write_buffer) を介して FPGA に送信されます。
 7. Flash から繰り返しデータをリードバックし、書き込まれたビットストリームファイルと比較します (API:recv_one_y_page_and_verify)。
 8. 繰り返しリードバック・比較が完了したら、autoboot-pattern データを最初の X-page に書き込みます。autoboot-pattern データは API (i2c_prog_xpage)によってパッケージ化され、MCU の I2C インターフェース (API : i2c_write_buffer)を介して FPGA に送信されます。
 9. ISC_DISABLE(0x3A)と ISC_NOOP(0x02)を送信します。この 2 つの命令は API (write_inst_and_data)によってパッケージ化され、MCU の I2C インターフェース(API: i2c_write_buffer) を介して FPGA に送信されます。

オンチップ Flash をプログラムするためのフローチャートを図 3-6 に示します。

図 3-6 オンチップ Flash のプログラミングのフローチャート



4 API の説明

4.1 よく使用される API のまとめ

表 4-1 よく使用される API のまとめ

名称	機能	備考
gw_i2c_read_code()	FPGAのCodeを読み出します	ID Code、Status Code、User Codeを読み出すことができます
gw_i2c_prog_flash_tsmc()	Em-Flashをプログラムします	-
gw_i2c_erase_flash_tsmc()	Em-Flashを消去します	
gw_i2c_send_reset()	IPをリセットします	
gw_i2c_read_version()	IPのバージョンを読み出します	

4.2 IP の基本的な命令

4.2.1 命令の定義

Gowin goConfig I2C IP 2C シリーズでサポートされている命令(コマンド)を表 4-2 に示します。

表 4-2 2C シリーズの命令の定義

番号	命令名	命令のアドレス(CMD)	説明	命令のフォーマット	サンプル・コードでの定義	備考
1	Status	0x44	ステータス・レジスタ読み出し命令	CMD	READ_STATUS_INST	-
2	Reset	0x55	IPリセット命令	CMD	IP_RESET_INST	
3	Send (data back)	0x66	データを送信してFlashからデータをリードバックする命令	CMD + Length + Data + Check_sum	SEND_DATA_B_ACK_INST	

番号	命令名	命令のアドレス(CMD)	説明	命令のフォーマット	サンプル・コードでの定義	備考
4	Waiting	0x77	待機命令	CMD + Length + Check_sum	WAITING_INST	
5	Send (No data back)	0x88	データを送信してFlashからデータをリードバックしない命令。	CMD + Length + Data + Check_sum	SEND_NO_DATA_BACK_INST	
6	Reconfig	0x99	Reconfig(再度コンフィギュレーション)命令	CMD	RECONFIG_N_INST	
7	IP Version	0xAA	IPバージョン命令	CMD	IP_VERSION_INST	-

命令フォーマットの定義および説明は次のとおりです。

- **CMD**: サポートされる命令については表 4-2 参照。
 - 定義: 命令のアドレス。
 - 幅: 8 ビット。
 - **Length**: 最小値は 2
 - 定義: 送信する有効バイト長^[1]。
 - 幅: 16 ビット。下位 8 ビット (Length_L) は上位 8 ビット (Length_H) より先に送信されます。
 - 計算式: Length = 有効データ長 + 2。
- 注記:
- ^[1] 有効バイト長は、有効データ (Data) のバイト数 + 2 です。例えば、0x66 命令の場合、Length = Data のバイト長 + 2。
 - 例えば、予想入力バイト長が 10 の場合、Length = 10 + 2 = 12 になります。
- **Data**:
 - 定義: 送信される有効データ。
 - 幅: 8 ビットで 1 データ。
 - **Length**: 送信データの実際の長さは Length-2 です。Data_0 は最初の有効データ、Data_1 は 2 番目の有効データ、Data_N は最後の有効データを示します。
 - **Check_sum**:
 - 定義: チェックサム
 - 幅: 16 ビット。上位 8 ビット (Check_sum_H) は下位 8 ビット (Check_sum_L) より先に送信されます。

ット (Check_sum_L) より先に送信されます。

- 計算式 : $\text{Check_sum} = \text{CMD} + \text{Length_L} + \text{Length_H} + \text{Data_0} + \dots + \text{Data_N}$ 。

注記 :

Check_sum がオーバーフローする場合、下位 16 ビットだけが留保され、残りは破棄されます。例えば、計算結果が Check_sum=0x10F11 の場合、Check_sum_H=0x0F、Check_sum_L=0x11 となり、残りは破棄されます。

Status(0x44)

ステータス・レジスタ読み出し命令。読み出し専用。読み出されるデータはステータス・レジスタの値です(表 4-3)。

表 4-3 ステータス・レジスタ

ビット	デフォルト値	説明	備考
Bit[7:3]	0	Reserved	-
Bit [2]	0	1 : Flashインターフェース動作中 0 : Flashインターフェースがアイドル	
Bit [1]	0	1 : チェックサムエラー 0 : チェックサムが正しい	
Bit [0]	0	1 : I ² Cステータスエラー 0 : I ² Cステータス正常	-

ステータス・レジスタ読み出し命令の例を図 4-1 に示します。

図 4-1 ステータス・レジスタ読み出し命令の例

S	Address	W	A	0x44	Sr	Address	R	A	Data	N	P
---	---------	---	---	------	----	---------	---	---	------	---	---

注記 :

以下の同様の図は、特に断りのない限り、この例と同じ意味です。

- 図中の「白いシェーディング」が「I²C マスター」から「I²C スレーブ」への送信、「灰色のシェーディング」が「I²C スレーブ」から「I²C マスター」への送信です。
- 図の中で使われる名前は、以下のように定義されています。
 - S : I²C バスの "開始(START)"。
 - Address : I²C バスの"スレーブアドレス (Slave Address)"。
 - W : I²C バスの"書き込み(WRITE)"。
 - A : I²C バスの"応答(ACK)"。
 - Sr : I²C バスの"再度開始(Repeated START)"。
 - R : I²C バスの"読み出し(READ)"。
 - Data : I²C バスで転送される有効データ。
 - N : I²C バスの"否定応答(NACK)"。
 - P : I²C バスの"停止(STOP)"。

- Length_L : Length の下位 8 ビット。
- Length_H : Length の上位 8 ビット。
- Data_0 : 最初の有効データ。
- Data_N : 最後の有効データ。
- Check_sum_H : Check_sum の上位 8 ビット。
- Check_sum_L : Check_sum の下位 8 ビット。

Reset(0x55)

IP リセット命令。IP の内部状態をリセットし、内部 FIFO をクリアします。

注記：

この処理は IP にのみ関係し、バックグラウンドアップグレードのプロセスには関係しません。

図 4-2 IP リセット命令の例

S	Address	W	A	0x55	A	P
---	---------	---	---	------	---	---

Send(0x66,data back)

データを送信して Flash からデータをリードバックする命令。I2C インターフェースは、リードバックされるデータのバイト数 (Length) と同時にデータを送信し、チェックサムが正しければ、IP は Flash インターフェースに変換されてデータを FIFO にリードバックします (図 4-3 参照)。このとき、ユーザー側でこのデータが必要であれば、I2C インターフェースを介して読み出し命令を送信して FIFO 内のデータを読み出すことができます (図 4-4 参照)。

簡単に説明すると、リードバック操作には 2 つのステップがあります。

1. 読み出されるデータを IP の FIFO に読み込みます。
2. FIFO のデータを I²C バス経由で読み出します。

注記：

- ステップ 1 とステップ 2 の間に、例えばステータスを読み出す命令などを挿入することができます。
- 図 4-3 では、Data_0~Data_N は、どのような値でも構いませんが、0x00 または 0xFF を書き込むことが推奨されます。
- 図 4-4 では、Data_0~Data_N は実際にリードバックされた値です。

図 4-3 リードバック命令の例

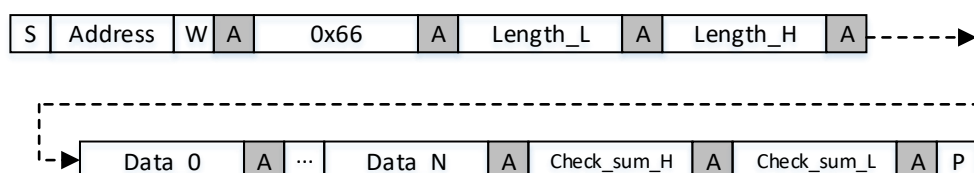
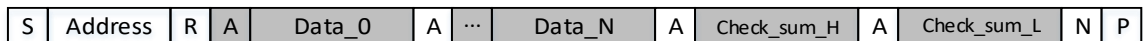


図 4-4FIFO データ読み出し命令の例



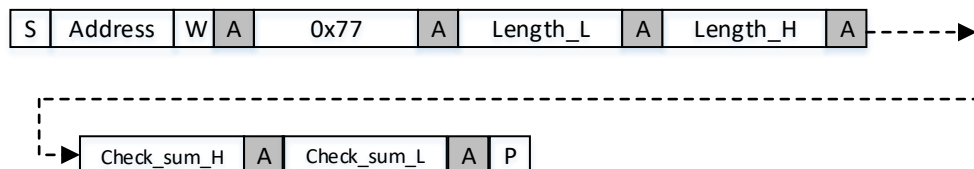
Waiting(0x77)

待機命令。I²C インターフェースは、待機バイト数（Length）を送信し、チェックサムが正しければ、「待機バイト数」のカウン트가完了するまで、Flash インターフェースはアイドルクロック（データは転送されない）を送信し続けることになります。

注記：

2C シリーズでは、Flash インターフェースの周波数は 2.5MHz で、すなわち 1 サイクルは 400ns です。1 バイトの場合は $8 \times 400 \text{ ns} = 3200 \text{ ns} = 3.2 \mu\text{s}$ 。例えば、150ms 待機する場合、待機バイト数は $46875 = 0xB71B$ となります。入力バイト数が $0xB71B + 2 = 0xB71D$ の場合、Length_L=0x1D、Length_H=0xB7 ということが分かります。

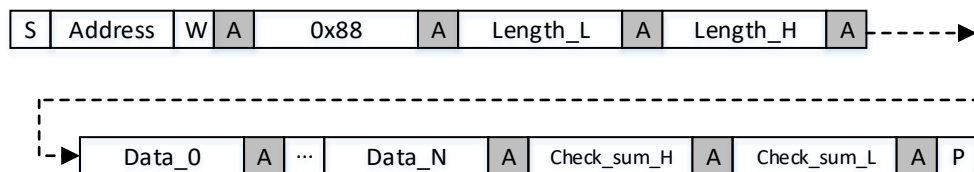
図 4-5 待機命令の例



Send(0x88,No data back)

データを送信して Flash からデータをリードバックしない命令。I²C インターフェースは、まず送信されるデータを FIFO に格納し、データカウン트가終了してチェックサムが正しい場合、FIFO 内のデータは Flash インターフェース経由で一括して送信されます。

図 4-6 データを送信して Flash からデータをリードバックしない命令の例



Reconfig(0x99)

Reconfig 命令。I²C インターフェースを介してこの命令を送信することにより、「Reconfig_N」ピンをトリガ(つまり、プルダウン)することができます。

図 4-7 Reconfig 命令の例



IP Version(0xAA)

IP バージョン命令。I²C インターフェースを介してこの命令を送信す

ると、固定値の **Data** が読み出されます。

注記：

現在のバージョンでは、Version_Data=0x20 です。

図 4-8 IP バージョン命令

S	Address	W	A	0xAA	Sr	Address	R	A	Data	N	P
---	---------	---	---	------	----	---------	---	---	------	---	---

4.3 コード読み出し API

4.3.1 API の紹介

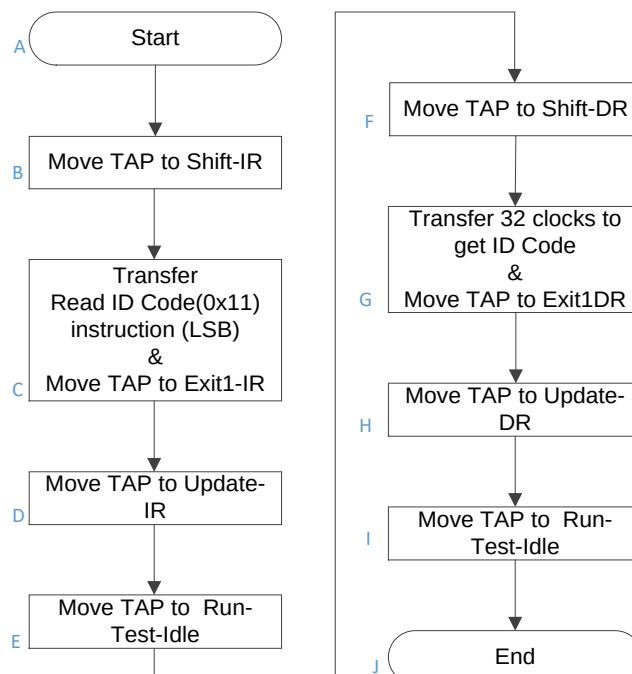
FPGA の ID Code、Status Code、User Code を読み出します。例：
gw_i2c_read_code(0x11)の場合、返されるコード値：0x10281B。

4.3.2 ID Code 読み出しの原理

下図は、JTAG による ID Code 読み出しの状態機械のフローチャートであり、各ステップにはそれぞれ A、B、C、D、E、F、G、H、I、J の番号が付けられています。IP で ID Code を読み出すには、下図に示す状態移遷用のパケットを構築してこのパケットを 0x88 命令および 0x66 命令を通じて IP に送信する必要があります。これに基づいて、IP は内部で JATG インターフェースに変換して ID Code の読み出しを実現します。

パケットの構造を図 4-9 に示します。

図 4-9 パケットの構造



4.3.3 I2C を介してコードを読み出す手順

I2C を介してコードを読み出す手順は次のとおりです。

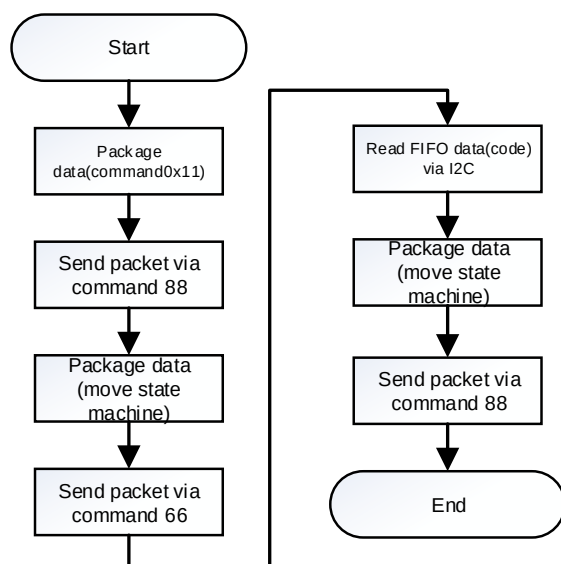
1. A、B、C、D、E、F 状態移遷データおよび 0x11 をパッケージ化しま

す。

2. **0x88** 命令を使用してパケットを送信します。**0x88** 命令の形式については、**4.2 IP** の基本的な命令を参照してください。
3. G、H 状態移遷データをパッケージ化します。
4. **0x66** 命令を使用してパケットを送信します。**0x66** 命令の形式については、**4.2 IP** の基本的な命令を参照してください。
5. I2C で FIFO 内のデータ(4 バイトのコード)を読み出します。
6. I、J 状態移遷データをパッケージ化します。
7. **0x88** 命令を使用してパケットを送信します。

I2C を介してコードを読み出すフローチャートは、図 4-10 に示しております。

図 4-10 I2C を介してコードを読み出すフローチャート



5 サンプル・コードの移植

5.1 コード・インターフェースの紹介

表 5-1 コード・インターフェースの紹介

インターフェースのタイプ	API	備考
上位層の機能インターフェース	gw_i2c_send_reset gw_i2c_send_reconfig_n gw_i2c_read_version gw_i2c_read_code gw_i2c_read_reg_status gw_i2c_erase_flash_tsmc gw_i2c_prog_flash_tsmc i2c_configmode_program_sram gw_i2c_send_runtest_xus match_code get_idcode_from_file	そのまま直接呼び出します。
	delay_ms delay_us	使用するプラットフォームに応じて変更する必要があります これらのインターフェースは、 delay.c ファイルにあります。
中間層インターフェース（データのパッケージ化、命令の組み合わせなど）	write_inst write_inst_and_data i2c_send_pack_tdi_tms write_move_tapdata write_data_at_tap_shiftdr i2c_rcv_pack_tdo i2c_jtag_waiting_count i2c_prog_xpage write_one_x_address write_one_y_page	そのまま直接呼び出します。
下位層のI2C読み出し書き込みインターフェース	i2c_init i2c_write_buffer i2c_write_buffer_2 i2c_write_byte	I2Cの初期化と基本的な読み出し書き込みを実現します。使用するプラットフォームに応じて変更する必要があります。

インターフェースのタイプ	API	備考
	i2c_read_buffer i2c_read_bufferxx	これらのインターフェースは、 bsp_i2c.cファイルにあります。

5.1.1 上位層の機能インターフェース

上位層の機能インターフェースは **main.c** で直接呼び出されます。上位層インターフェースは、いくつかの機能を実装するために中間層インターフェースを呼び出します。上位層インターフェースは、いくつかのシンプルな機能を実装するために下位層の読み出し書き込みインターフェースを直接呼び出します。

5.1.2 中間層インターフェース

中間層インターフェースは主にデータの組み合わせとパッケージ化を実装します。中間層は、パッケージ化されたデータを送信するために下位層の読み出し書き込みインターフェースを呼び出します。

5.1.3 下位層の I2C 読み出し書き込みインターフェース

下位層の読み出し書き込みインターフェースは、上位層の機能インターフェースと中間層のインターフェースによって呼び出されるため、この下位層のインターフェースの名前は変更しないことをお勧めします。

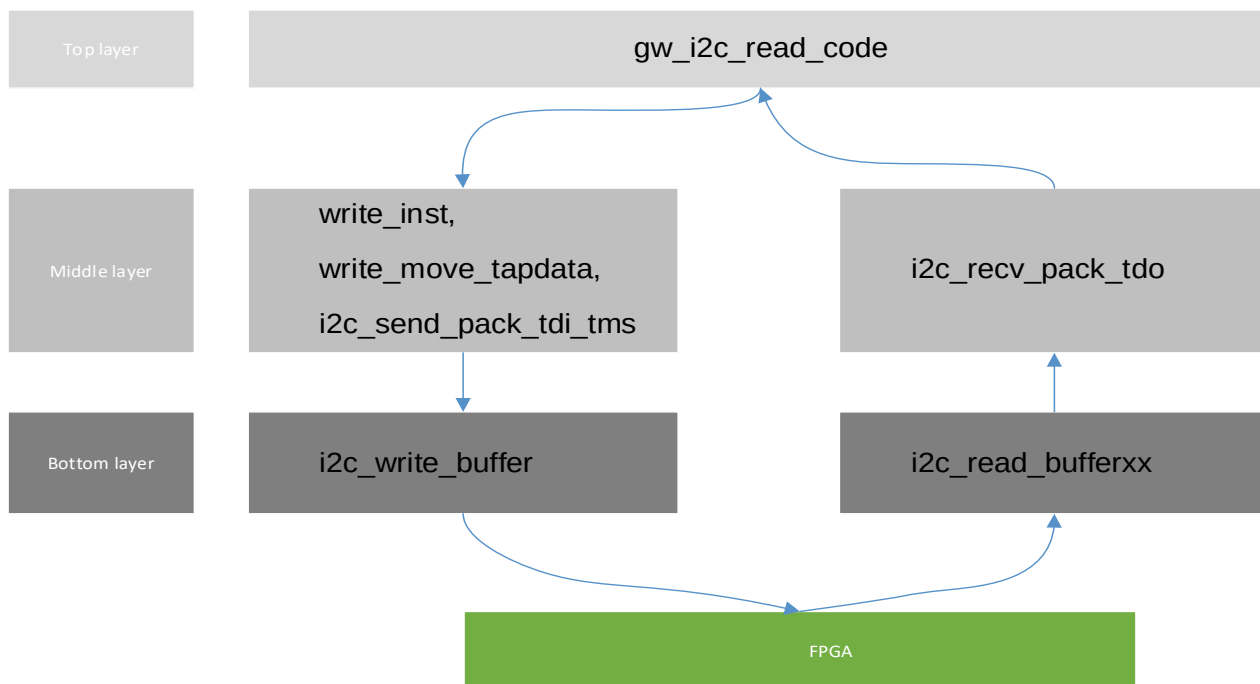
i2c_init、**i2c_read_buffer**、**i2c_read_bufferxxx**、**i2c_write_buffer** など、対応する関数内で I2C の初期化、読み出し、書き込みを実装するだけでよいです。また、**i2c_write_buffer_2** インターフェースが **FPGA** 内の専用のハードウェア・コンフィギュレーション・モジュールを通じて **SRAM** をコンフィギュレーションするために使用されることに注意してください。

goConfig IP-I2C は、標準の I2C プロトコルをサポートしており、I2C の読み出しおよび書き込みプロセスは固定ステップです。

私たちが提供するサンプル・コードの下位層インターフェースには 2 つのモードがあります。1 つのモードでは **MCU** のハードウェア I2C が使用され、もう 1 つのモードでは **GPIO** を使用して I2C をエミュレートします。マクロ **USE_HARDWARE_I2C** を使用して I2C モードを選択します：**#define USE_HARDWARE_I2C** の場合、ハードウェア I2C が使用されます。

5.1.4 インターフェースの呼び出し関係

図 5-1 インターフェースの呼び出し関係



5.2 下位層インターフェースの書き換えの説明

5.2.1 i2c_init

i2c_init 関数インターフェースは主にピンの定義、I2C モードの設定、および I2C レートの設定を実装しており、使用するプラットフォームに応じて書き換える必要があります。

例えば：

```

void i2c_init(void)
{
    I2C_GPIO_Config();
    I2C_Mode_Config();
}
  
```

次の説明では、たとえば次のような略語が使用されています。

- i2c start : Start
- i2c stop : Stop
- fpga device IP address : ADDRESS（例えば、アドレスは 7bit : 1011000 (0x58)）
- write control bit : WR
- read control bit : RD

- ACK のチェック : check ACK
- Command Address : CMD ADDRESS

5.2.2 i2c_write_buffer

この関数インターフェースは主に I2C のデータ書き込みを実装し、使用するプラットフォームに応じて書き換える必要があります。次の手順を実行して I2C データ書き込み機能を実現します。

その手順は次のとおりです：

1. Start
2. send (ADDRESS<<1) | WR
3. check ACK
4. send CMD ADDRESS
5. check ACK
6. send n byte data
7. check ACK (1 バイトのデータを送信するたびに check ACK が必要です)
8. Stop

5.2.3 i2c_read_buffer

この関数インターフェースは主に I2C のデータ読み出し(主に **status register** 読み出し用の **0x44** 命令と **IP Version** 読み出し用の **0xAA** 命令に使用される)を実装し、使用するプラットフォームに応じて書き換える必要があります。次の手順を実行して I2C 読み出し機能を実現します。

1. Start
2. send (ADDRESS<<1) | WR
3. check ACK
4. send CMD ADDRESS
5. check ACK
6. Start
7. send (ADDRESS<<1) | RD
8. check ACK
9. i2c receive n byte data
10. NACK (最後の 1 バイトの後に NACK を送信する)
11. Stop

5.2.4 i2c_read_bufferxx

この関数インターフェースは主に I2C のデータ読み出し(主に **0x66** 命令に使用される)を実装し、使用するプラットフォームに応じて書き換える必要があります。次の手順を実行して I2C 読み出し機能を実現します。

1. Start
2. send (ADDRESS<<1) | RD
3. check ACK
4. i2c receive n byte data
5. NACK (最後の 1 バイトの後に NACK を送信する)
6. Stop

注記：

i2c_read_bufferxx と i2c_read_buffer の違いは、i2c_read_bufferxx で send ADDRESS | WR ステップが省略されていることです。linux kernel ライブラリを使用している場合は、これを追加する必要がある場合があります。

5.3 移植のポイント

5.3.1 準備

1. 対象プラットフォームで実行できる基本的な I2C 読み出しおよび書き込みプロジェクト(**basic_i2c_project**)を準備します。
2. I2C 読み出しおよび書き込みプロジェクト(**basic_i2c_project**)でマルチバイトの書き込みを実現できる関数をカプセル化します：

- そのパラメータと名前 : `uint32_t i2c_write_buffer(u8 *pBuffer, u8 WriteAddr, u32 NumByteToWrite)` (5.2 下位層インターフェースの書き換えの説明を参照)。
 - マルチバイトの読み出しを実現できる関数をカプセル化します : `uint32_t i2c_read_buffer(u8 *pBuffer, u8 ReadAddr, u16 NumByteToRead)`(5.2 下位層インターフェースの書き換えの説明を参照)。
 - マルチバイトの読み出しを実現できる関数をカプセル化します : `uint32_t i2c_read_bufferxx(u8 *pBuffer, u8 ReadAddr, u16 NumByteToRead)`(を参照)。
3. サンプル・コードを用意します。主に、`main.c`、`i2c_jtag.c`、`i2c_jtag.h`、`bsp_i2c.h`、`bsp_i2c.c` が使用されます。
 4. 上位層と中間層の関数インターフェースを対象プラットフォームの `basic_i2c_project` プロジェクトにコピーします。

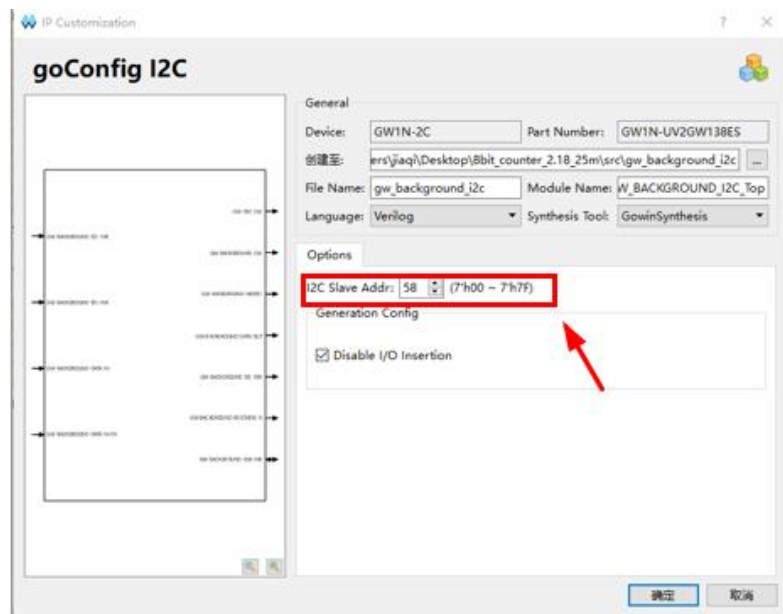
5.3.2 コンパイルと検証

1. 移植が完了したら、まずコンパイルします。コンパイルが正しく、警告やエラーがないことを確認します。
2. 対象プラットフォームを **FPGA** の **I2C** に接続します。
3. `gw_i2c_send_reset` インターフェースと `gw_i2c_read_version` インターフェースを実行します。バージョン番号を読み出すことができれば、動作が正常であることが分かります。

5.3.3 トラブルシューティング

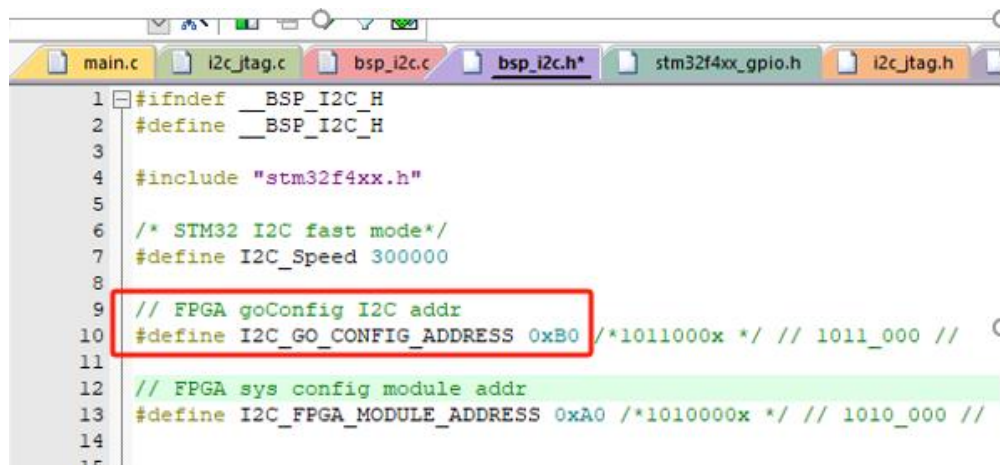
1. インターフェース `gw_i2c_read_version` がバージョン番号を読み出せない場合 :
 - a) 配線が正常かどうか、**SDA**、**SCL**、**GND** の接続が良好かどうかを確認する必要があります。
 - b) オシロスコープまたはロジック・アナライザを使用して、**MCU** の **I2C** から信号の出力があるかどうかを観察します。
 - c) **FPGA** に `goConfig` IP が実装されているかどうかを確認します。実装されていない場合、`goConfig` IP を **FPGA** にダウンロードする必要があります。
2. デバイスアドレスが送信された後に **ACK** がない場合 :
 - a) 設定された `goConfigIP` のアドレスと一致しているかどうかを確認します : The i2c address of goconfigIP is set to 0x58 (1011 000)。

図 5-2 アドレスの確認



- b) Cコードでは、マクロは次のように定義されます：`#define I2C_GO_CONFIG_ADDRESS 0xB0` (1011 0000。0xB0は0x58を1ビット左にシフトしたものです($0xB0 = 0x58 \ll 1$)。組み込みシステムでは、7ビットも8ビットとして処理されます。

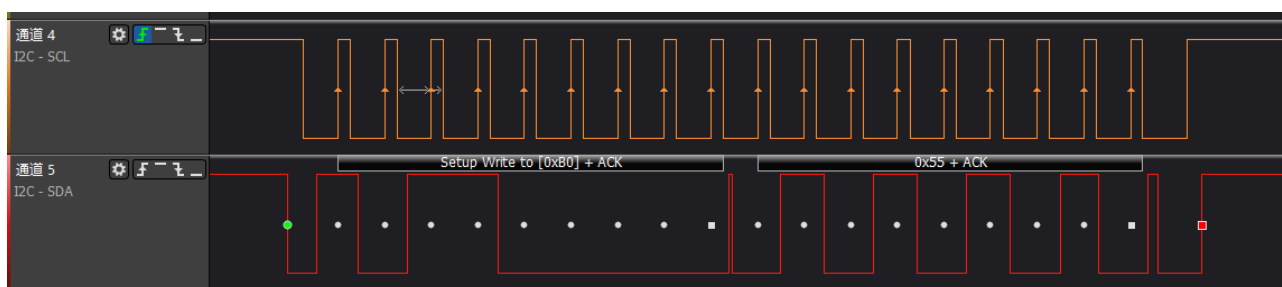
図 5-3 マクロの定義



5.3.4 参考タイミング図

下図は、`gw_i2c_send_reset`を実行する際のタイミングです (7ビットのデバイスアドレス 1011000 を送信後、命令 0x55 を送信)。

図 5-4 タイミング図



下図は、`gw_i2c_read_reg_status` を実行する際のタイミングです
(命令 `0x44` を送信後、データ `0x00` をリードバック)。

図 5-5 タイミング図

