Програмування по інтерфейсу JTAG

1 Використання інтерфейсу JTAG для програмування кристала

Інтерфейс JTAG був розроблений групою провідних фахівців з проблем тестування електронних компонентів (Joint Test Action Group). Вбудований у більшість мікроконтролерів набору Mega, інтерфейс JTAG може бути використаний для:

– тестування друкованих плат;
– конфігурації (програмування) кристала;
– внутрішньосхемного налагодження.

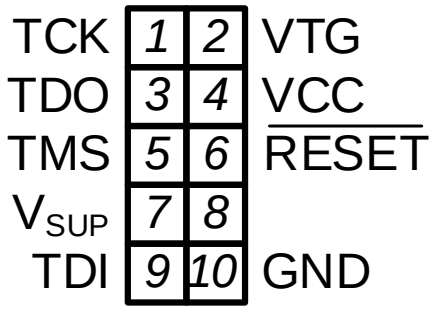

Рисунок 3.4 – Розводка роз'єму JTAG

Доступ до модуля JTAG здійснюється через чотири виводи мікроконтролера, що складають так званий «порт тестового доступу» (Test Access Port – TAP): TMS, TCK, TDI і TDO. Відповідність цих виводів контактам портів введення/виведення мікроконтролера, а також їхні функції наведені в табл. 3.19. Стандартний роз'єм для підключення пристроїв наведений на рис. 3.4.

Роботою модуля JTAG керує TAP-контролер, який є скінченним автоматом зі 16 станами. Перехід між станами здійснюється за наростаючим фронтом сигналу TCK відповідно до сигналу, що присутній на виводі TMS. Після ввімкнення живлення контролер знаходиться в стані Test-Logic-Reset.

Дозвіл/заборона інтерфейсу JTAG здійснюється за допомогою конфігураційної комірки JTAGEN. Якщо вона не запрограмована (1), то виводи TAP працюють як звичайні контакти портів введення/виведення, а TAP-контролер знаходиться в стані скидання. Для ввімкнення інтерфейсу комірка JTAGEN має бути запрограмована (стан за замовчуванням). До того ж, має бути скинутий біт JTD регістра MCUCSR або MCUCR (рис. 3.5). Причому, для зміни стану цього біта нове значення необхідно записати в нього двічі протягом чотирьох тактів.

Таблиця 3.19 – Виводи, які використовуються інтерфейсом JTAG

Назва лінії інтерфейсу	ATmega16x/32x	ATYnega64x/128x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega 325x/3250x	ATmega640x/1280x/2560x	Призначення виводів
TCK	PC2	PF4	PC4	PC2	PF4	PF4	PF4	Вхід тактового сигналу
TMS	PC3	PF5	PC5	PC3	PF5	PF5	PF5	Вхід вибору режиму
TDO	PC4	PF6	PC6	PC4	PF6	PF6	PF6	Вихід даних
TDI	PC5	PF7	PC7	PC5	PF7	PF7	PF7	Вхід даних

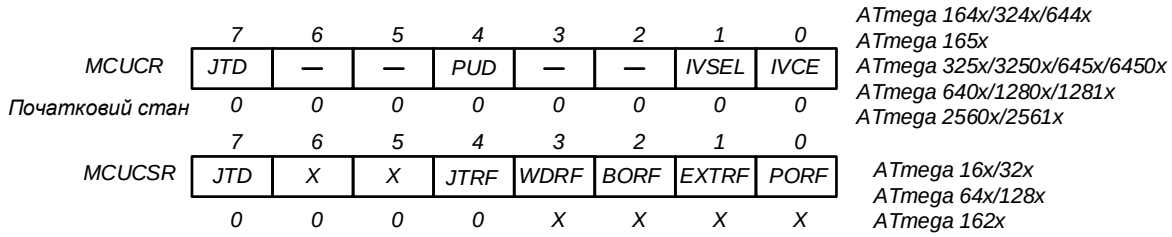

Рисунок 3.5 – Регістри MCUCSR/MCUCR
щодо інтерфейсу JTAG

Команди JTAG для програмування

З 16 команд, підтримуваних інтерфейсом, при програмуванні використовуються тільки п'ять. Описи цих команд наведені нижче.

AVR_RESET (код команди $ 0C). Ця команда призначена для переведення мікроконтролера в стан скидання і відповідно виведення його з цього стану. Як регістру даних обирається 1-бітний регістр скидання (Reset Register). Запис «1» у цей регістр еквівалентна подачі на вивід RESET мікроконтролера напруги рівня «0». У стані скидання мікроконтролер буде перебувати до тих пір, поки в регістр скидання не буде записаний «0».

Активний стан: Shift-DR – здійснюється завантаження регістра скидання.

PROG_ENABLE (код команди $ 04). Ця команда призначена для дозволу програмування кристала через порт JTAG. Як регістр даних обирається 16-бітовий регістр дозволу програмування (Programming Enable Register). При запису в цей регістр числа $A370 (сигнатура дозволу програмування) дозволяється програмування мікроконтролера по інтерфейсу JTAG. При виході з режиму програмування цей регістр має скидатися.

Активні стани:

– Shift-DR – здійснюється завантаження регістра дозволу переривання;

– Update-DR – здійснюється порівняння вмісту регістра з числом $A370 і в разі збігу – переведення мікроконтролера в режим програмування.

PROG_COMMANDS (код команди $ 05). Ця команда призначена для завантаження команд програмування і видачі результатів їхнього виконання (якщо вони є). Як регістр даних обирається 15-бітний регістр команд (Programming Command Register).

Активні стани:

– Capture-DR – результат виконання попередньої команди завантажується в регістр;

– Shift-DR – за наростаючим фронтом сигналу TCK здійснюється видача результату з одночасним завантаженням нової команди;

– Update-DR – завантажена команда подається на блок пам'яті мікроконтролера;

– Run-Test/Idle – генерується один тактовий імпульс, необхідний для виконання команди.

PROG_PAGELOAD (код команди $06). Ця команда призначена для безпосереднього завантаження сторінки пам'яті програм через порт JTAG. Реалізація цієї команди залежить від моделі мікроконтролера. У «старих» моделях (ATmegal6x/32x/64x/128x і ATmegal62x) як регістр даних обирається регістр завантаження віртуальної сторінки (Virtual FLASH Page Load Register), розмір якого дорівнює розміру однієї сторінки пам'яті програм. Власне зсувний регістр є 8-бітовийм, а пересилання по байтах даних в буфер здійснюється автоматично.

Активний стан: Shift-DR – здійснюється побітова завантаження даних і побайтне їхнє пересилання в буфер сторінки FLASH-пам'яті.

В інших моделях як регістр даних обирається 8-бітовий регістр даних FLASH-пам'яті. Активні стани:

– Shift-DR – здійснюється побітне завантаження байта даних;

– Update-DR – вміст регістра даних FLASH-пам'яті копіюється у тимчасовий регістр в буфері сторінки. Чергування старшого і молодшого байтів при кожному проходженні стану Update-DR здійснюється автоматично, починаючи з молодшого байта при завантаженні команди.

PROG_PAGEREAD (код команди $07). Ця команда призначена для зчитування вмісту сторінки пам'яті програм через порт JTAG. Реалізація цієї команди також залежить від моделі мікроконтролера. У «старих» моделях (ATmcgal6x/32x/64x/128x і ATmegal62x) як регістр даних обирається регістр читання віртуальної сторінки (Virtual FLASH Page Read Register), розмір якого на 1 байт більше розміру сторінки пам'яті програм. Побайтне пересилання вмісту сторінки у зсувний регістр здійснюється автоматично.

Активні стани: Shift-DR – здійснюється побайтне пересилання вмісту сторінки FLASH-пам'яті у зсувний регістр і побітова його видача на вивід TDO. Перші 8 тактів використовуються для первинного завантаження зсувного регістру, оскільки біти, що вивантажуються в цей час, необхідно ігнорувати.

В інших моделях як регістр даних обирається 8-бітний регістр даних FLASH-пам'яті. Активні стани:

– Capture-DR – вміст обраного байта замикається в регістрі даних FLASH-пам'яті. Чергування старшого і молодшого байтів при кожному проходженні стану Capture-DR здійснюється автоматично, починаючи з молодшого байта при завантаженні команди.

– Shift-DR – здійснюється побітове завантаження байта даних.

3 Алгоритм програмування

У розділі описуються дії, які необхідно виконувати для програмування мікроконтролерів через порт JTAG. Формат усіх команд, які використовуються при програмуванні, наведено в табл. 3.20.

Таблиця 3.20 – Команди програмування по інтерфейсу JTAG

Команда		TDI	TDO	Примітка
1	a. Знищення кристала	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	b. Запит стану операції	0110011_10000000	xxxxx0x_xxxxxxxx	2
2	a. Вхід у режим запису FLASH-пам'яті	0100011_00010000	xxxxxxx_xxxxxxxx	–
	b. Завантаження додаткового байта адреси	0001011_cccccccc	xxxxxxx_xxxxxxxx	–
	c. Завантаження старшого байта адреси	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	–
	d. Завантаження молодшого байта адреси	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	e. Завантаження молодшого байта даних	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	–
	f. Завантаження старшого байта даних	0010111_iiiiiiii	xxxxxxx_xxxxxxxx	–
	g. Фіксація даних	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	h. Запис сторінки FLASH-пам'яті	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	-

Продовження таблиці 3.20

Команда		TDI	TDO	Примітка
	i. Запит стану операції	0110111_00000000	xxxxxxx_xxxxxxxx	2
3	a. Вхід в режим читання FLASH-пам'яті	0100011_00000010	xxxxxxx_xxxxxxxx	–
	b. Завантаження додаткового байта адреси	0001011_cccccccc	xxxxxxx_xxxxxxxx	–
	c. Завантаження старшого байта адреси	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	–
	d. Завантаження молодшого байта адреси	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	e. Читання молодшого і старшого байтів даних	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_00000000 xxxxxxx_00000000	9 8
4	a. Вхід в режим запису EEPROM-пам'яті	0100011_00010001	xxxxxxx_xxxxxxxx	–
	b. Завантаження старшого байта адреси	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	–
	c. Завантаження молодшого байта адреси	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	d. Завантаження байта даних	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	–
	e. Фіксація даних	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx xxxxxxxx xxxxxxx xxxxxxxx xxxxxxx xxxxxxxx	–
	f. Запис сторінки EEPKOM-пам'яті	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	g. Запит стану операції	0110011_00000000	xxxxx0x_xxxxxxxx	2
5	a. Входу режим читання EEPROM-пам'яті	0110011_00000011	xxxxxxx_xxxxxxxx	–
	b. Завантаження старшого байтуадреси	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	–
	c. Завантаження молодшого байта адреси	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	d. Читання байта даних	0110011_bbbbbbbb 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	–

Продовження таблиці 3.20

Команда		TDI	TDO	Примітка
6	a. Вхід у режим запису конфігураційних комірок	0100011_01000000	xxxxxxx xxxxxxxx	–
	b. Завантаження байта даних	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	3,5
	c. Запис додаткового конфігураційного байта	0111011_00000000 0111001_00000000 0111011_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	d. Запит стану операції	0110111_00000000	xxxxxox_xxxxxxxx	2
	e. Завантаження байта даних	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	3,5
	f. Запис старшою конфігураційного байта	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	g. Запит стану операції	0110111_00000000	xxxxxxx_xxxxxxxx	2
	h. Завантаження байта даних	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	3, 5
	i. Запис молодшого конфігураційного байта	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
6	j. Запит стану операції	0110011_00000000	xxxxxxx_xxxxxxxx	2
7	a. Вхід у режим запису комірок захисту	0100011_00100000	xxxxxxx_xxxxxxxx	–
	b. Завантаження байта даних	001001_11iiiiii	xxxxxxx_xxxxxxxx	3,6
	c. Запис байта зашитий	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–
	d. Запит стану операції	0110011_00000000	xxxxxxx_xxxxxxxx	2
8	a. Вхід у режим читання конфігураційних комірок і комірок захисту	0100011_00000100	xxxxxxx_xxxxxxxx	–
	b. Читання додаткового конфігураційного байта	0111010_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	5
	c. Читання старшого конфігураційного байта	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	5

Продовження таблиці 3.20

Команда		TDI	TDO	Примітка
	d. Читання мол. конфігураційного байта	0110010_00000000 0110011_00000000	xxxxxxx xxxxxxxx xxxxxxx_oooooooo	5
	e. Читання байта захисту	0110110_00000000 0110111_00000000	xxxxxxx xxxxxxxx xxxxxxx xxoooooo	4, 6
	f. Читання конфігураційних комірок і комірок захисту	0111010 00000000 0111110 00000000 0110010 00000000 0110110 00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxxoooooooo xxxxxxx oooooooo xxxxxxx oooooooo xxxxxxx xxoooooo	4 4, 7 4, 8 4, 9 4, 10
9	a. Вхід у режим читання комірок ідентифікатора	0100011 00001000	xxxxxxx_xxxxxxxx	–
	b. Завантаження байта адреси	000001_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	c. Читання байта ідентифікатора	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	–
10	a. Вхід до режиму читання комірки калібрування	0100011_00001000	xxxxxxx_xxxxxxxx	–
	b. Завантаження байта адреси	000001_bbbbbbbb	xxxxxxx_xxxxxxxx	–
	c. Читання константи калібрування	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxxoooooooo	–
11	a. Завантаження команди «Немає операції»	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	–

Примітки:

1. Умовні позначення, що використовуються в таблиці:
a – біти старшого байта адреси;
b – біти молодшого байта адреси;
i – дані, що посилаються до мікроконтролера;
o – дані, що зчитуються з мікроконтролера;
x – довільний стан біта.
2. Повторювати до тих пір, поки o = 1.
3. Для програмування комірки відповідний їй біт має бути скинутий, для знищення – встановлений.
4. 0 – комірка запрограмована, 1 – незапрограмована.
5. Відповідність біт коміркам конфігурації (табл. 3.5).
6. Відповідність біт коміркам захисту (рис. 3.5).
7. Додатковий байт.
8. Старший байт.
9. Молодший байт.
10. Байт захисту.

Операції програмування (номера команд відповідно до табл. 3.20). Операція «Вхід до режиму програмування»:

1. Завантажити команду AVR_RESET і занести «1» у регістр скидання.

2. Завантажити команду PROG_ENABLE і занести в регістр дозволу програмування значення 1010_0011_0111_ 0000 ($ A370).

Операція «Вихід з режиму програмування»:

1. Завантажити команду PROG_COMMANDS.
2. Заборонити виконання всіх команд програмування, запустивши команду «Немає операції».
3. Завантажити команду PROG_ENABLE і занести в регістр дозволу програмування значення 0000_0000_0000_0000.
4. Завантажити команду AVR_RESET і занести «0» у регістр скидання.

Операція «Знищення кристала»:

1. Завантажити команду PROG_COMMANDS.
2. Почати знищення кристала, запустивши команду 1a.
3. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 1b. Як альтернативу можна просто зачекати (див. табл. 3.16).

Операція «Програмування FLASH-пам'яті»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим програмування FLASH-пам'яті (команда 2a).
3. Завантажити додатковий байт адреси (команда 2b).
4. Завантажити старший байт адреси (команда 2c).
5. Завантажити молодший байт адреси (команда 2d).
6. Завантажити дані (команди 2e, 2f і 2g).
7. Повторити п. 5 та 6 для кожної комірки сторінки пам'яті програм.
8. Зберегти сторінку (команда 2h).
9. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 2h.
10. Повторити п. 3...9 для програмування інших сторінок пам’яті програм.

Операція «Читання FLASH-пам'яті»:

1. Завантажити команду PROG_COMMANDS.
2. Перейти в режим читання FLASH-пам'яті (команда 3a).
3. Завантажити адресу комірки пам'яті (команди 3b, 3c та 3d).
4. Прочитати вміст комірки (команда 3c).
5. Повторити п. 3 та 4 для читання інших комірок пам'яті програм.

Операція «Програмування EEPROM-пам'яті»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим програмування EEPROM-пам'яті (команда 4a).
3. Завантажити старший байт адреси (команда 4b).
4. Завантажити молодший байт адреси (команда 4c).
5. Завантажити дані (команди 4d і 4e).
6. Повторити п. 4 і 5 для всіх комірок сторінки.
7. Зберегти сторінку (команда 4f).

8. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 4g.

9. Повторити п. 3...8 для програмування інших сторінок EEPROM-пам'яті.

Операція «Читання EEPROM-пам'яті»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим читання EEPROM-пам'яті (команда 5a).
3. Завантажити адресу комірки пам'яті (команди 5b та 5c).
4. Зчитати вміст комірки (команда 5d).
5. Повторити п. 3 та 4 для читання інших комірок EEPROM-пам'яті.

Операція «Програмування комірок конфігурації»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим програмування конфігураційних комірок (команда 6-a).
3. Завантажити байт даних (команда 6b).
4. Записати додатковий байт конфігурації (команда 6c).
5. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 6d.
6. Завантажити байт даних (команда 6e).
7. Записати старший байт конфігурації (команда 6f)
8. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 6g.
9. Завантажити байт даних (команда 6h).
10. Записати молодший байт конфігурації (команда 6i).
11. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 6j.

Операція «Програмування комірок захисту»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим програмування комірок захисту (команда 7a).
3. Завантажити байт даних (команда 7b).
4. Записати байт захисту (команда 7c).
5. Дочекатися закінчення виконання цієї операції, контролюючи значення, що повертається командою 7d.

Операція «Читання конфігураційних комірок та комірок захисту»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим читання конфігураційних комірок та комірок захисту (команда 8a).
3. Для читання всіх комірок виконати команду 8f.
4. Для читання тільки додаткового байта конфігурації виконати команду 8b.

5. Для читання тільки старшого байта конфігурації виконати команду 8c.

6. Для читання тільки молодшого байта конфігурації виконати команду 8d.

7. Для читання тільки байту захисту виконати команду 8e.

Операція «Читання комірок ідентифікатора»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим читання комірок ідентифікатора (команда 9a).
3. Завантажити адресу $00, використовуючи команду 9b.
4. Прочитати перший байт ідентифікатора (команда 9c).
5. Повторити п. 3 та 4 для читання другого (адреса $01) та третього (адреса $02) байта ідентифікатора.

Операція «Читання комірок калібрування»:

1. Завантажити команду PROG_COMMANDS.
2. Увійти в режим читання комірок калібрування (команда 10a).
3. Завантажити адресу комірки (команда 10b).
4. Зчитати значення константи калібрування (команда 10c).