Examen CELECT7 :

Applications des microcontrôleurs

mercredi 13 novembre 2019
13h30-15h00
Supports de cours et notes manuscrites autorisés

A- Questions de cours

(4pts)

1. Pourquoi ne faut-il pas utiliser les fonction arduino `delay()` et `millis()` dans une routine d'interruption ?
2. On souhaite mettre en oeuvre une liaison UART entre un microcontrôleur et un périphérique. Expliquer en quoi la solution basée sur la bibliothèque arduino `SoftwareSerial` est différente de la solution basée sur la classe `Serial` d'arduino.
3. Vrai ou Faux : l'interface TWI du microcontrôleur ATmega328p comporte un registre 8 bits pour l'émission et un registre 8 bits pour la réception.
4. Qu'est-ce qu'un capteur 9-DoF ? Que veut dire DoF ?
5. Pourquoi n'est-il pas possible d'utiliser sur un même lieu des canaux WiFi contigus (ex : canaux 2 et 3) ?
6. Combien de machines peuvent-elles se connecter à un réseau IP d'adresse 192.168.2.0 et de masque 255.255.255.224 ?
7. VRAI ou FAUX : l'esp8266 est un SoC construit autour d'un processeur de type microcontrôleur.
8. Avec la bibliothèque arduino esp8266WiFi, quelles sont les trois modes de configurations possibles d'un esp8266 ?

B- Manipulations binaires

1. (1,5pts) Soient `a` et `b` deux variables de type `char`. Quelle fonction est réalisée par l'ensemble des trois lignes de code suivantes :

```
a ^= b;
b ^= a;
a ^= b;
```

2. (2pts) Soit les variables entières signées `a` et `b`. Écrire en une ligne le code permettant de mettre dans la variable `b` les 4 bits de poids faible de la variable `a`.

C- Configuration des registres de l'ATmega 328p

1. (1pt) Timer2 - En vous appuyant sur les éléments de la Fig. 1, donner les lignes de code permettant de configurer le Timer2 en mode Fast PWM et que le timer compte jusque 0xFF avant de repasser à 0x00. Seuls les bits requis des registres TCCR2A et TCCR2B devront être modifiés, et leurs noms devront apparaître dans les lignes de code.

17.11.1 TCCR2A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
(0xB0)	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

17.11.2 TCCR2B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
(0xB1)	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	W	W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 17-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

FIGURE 1 – Éléments relatifs au Timer 2 de l'ATmega328p.

2. (3pts) Ports d'E/S - Ré-écrire le code ci-après en utilisant des fonctions de la couche d'abstraction arduino de façon à ce que les noms des registres DDRD, PORTD et PIND de l'ATmega328p n'apparaissent plus dans le code. On suppose que la carte arduino est une carte uno. Vous pouvez vous appuyer sur la Fig. 2 pour visualiser la correspondance entre les numéros de broches ATmega 328p et les numéros de broches de la carte arduino uno.

```
void setup() {
        Serial.begin(115200);
        DDRD = 96;
        PORTD = 160;
        delay(1);
        Serial.println(PIND>>5,BIN);
}
void loop() {
}
```

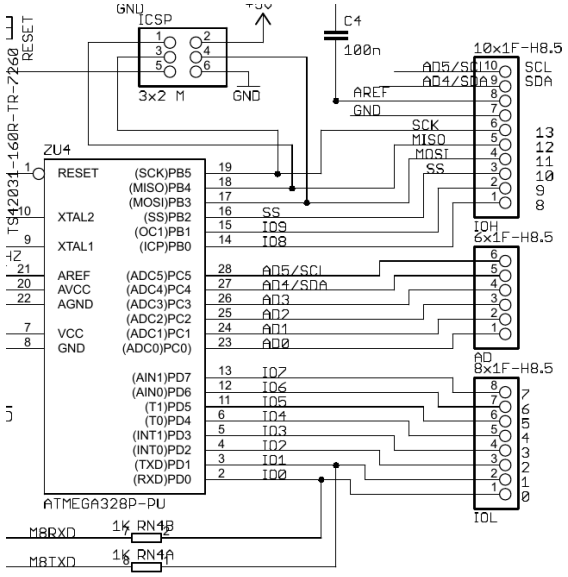
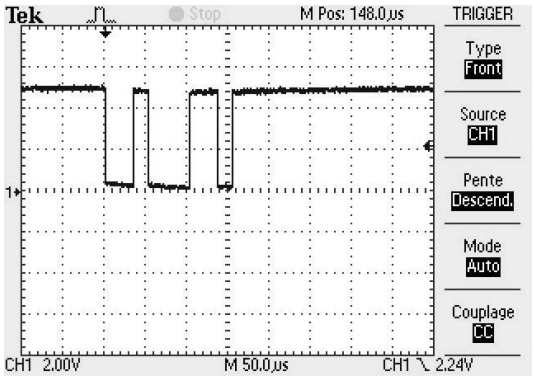


FIGURE 2 – Schéma partiel de la carte arduino uno.

3. (2pts) USART - Les registres qui permettent de configurer l'USART du microcontrôleur ATmega 328p sont UBRR0, UCSR0A, UCSR0B et UCSR0C.
   En vous appuyant sur les documents fournis en annexe 1, donner les lignes de code mettant en oeuvre ces registres afin que l'USART soit configurée de la manière suivante :
   — mode asynchrone
   — débit : 9600 bauds
   — trame : 6 bits de donnée, 1 bit de parité paire, 2 bits de stop
   Faire apparaître explicitement les noms des bits modifiés dans le code.

D- Liaisons série : UART et TWI/I2C

1. UART - Soit ci-après le relevé à l'oscilloscope de la ligne TX correspondant une commande arduino d'envoi d'un octet de données sur la liaison UART :



   A toutes fins utiles, la table des caractères ascii est rappelée en Fig. 3.
   (3pts) Proposer plusieurs alternatives de commandes arduino qui permettent d'envoyer ces données sur la lignes :
   (a) une commande avec la foncion `write` en supposant que l'option `SERIAL_8N1` a été passée à `Serial.begin()`
   (b) une autre commande avec la même fonction et la même option (fonction `write`, option `SERIAL_8N1` passée à `Serial.begin()`)
   (c) une commande avec la fonction `print` en supposant que l'option `SERIAL_8N1` a été passée à `Serial.begin()`

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

FIGURE 3 – Table des caractères ascii.

(d) une commande avec la fonction `write` en supposant que soit l'option `SERIAL_8E1` soit l'option `SERIAL_8O1` a été passée à `Serial.begin()` (précicer dans votre réponse à quelle option elle correspond)

2. TWI/I2C - Soit le code arduino ci-dessous permettant au microcontrôleur de récupérer une donnée `data` du capteur MPU9250 :

```
1  #include <Wire.h>
2  int16_t data;
3  void setup() {
4          Wire.begin();
5          Wire.beginTransmission(0x69);
6          Wire.write(0x6B);
7          Wire.write(0x00);
8          Wire.endTransmission();
9  }
10 void loop() {
11         Wire.beginTransmission(0x69);
12         Wire.write(0x45);
13         Wire.endTransmission(false);
14         Wire.requestFrom(0x69,2);
15         data = (Wire.read()<<8);
16         data |= Wire.read();
17 }
```

(a) (0,5pt) À quoi sert la ligne 4 du code ?

(b) (0,5pt) 'A quoi sert la trame générée par les lignes 5 à 8 du code ?

(c) (1pt) Écrire la structure de la trame I2C correspondant aux lignes 11 à 16 du code (faire apparaître les noms des champs parmi S, Sr, P, ACK, NACK, SLA+R, SLA+W, DATA, et souligner ceux qui sont transmis par le maître).

Pourquoi l'option `false` a-t-elle été utilisée à la ligne 13 ?

(d) (1,5pt) Écrire sans utiliser l'abstraction arduino le code correspondant aux lignes de code 11 à 13. Vous pouvez vous aider des documents fournis en annexe 2.

Annexe 1 : USART de l'ATmega328p

Table 19-12. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

	f_{osc} = 16.0000MHz			
	U2Xn = 0		U2Xn = 1	
Baud Rate (bps)	UBRRn	Error	UBRRn	Error
2400	416	–0.1%	832	0.0%
4800	207	0.2%	416	–0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	–0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	–0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	–0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	–3.5%	16	2.1%
230.4k	3	8.5%	8	–3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max.(1)	1Mbps		2Mbps	

Note: 1. UBRRn = 0, error = 0.0%

19.10 Register Description

19.10.1 UDRn – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The USART transmit data buffer register and USART receive data buffer registers share the same I/O address referred to as USART data register or UDRn. The transmit data buffer register (TXB) will be the destination for data written to the UDRn register location. Reading the UDRn register location will return the contents of the receive data buffer register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the transmitter and set to zero by the receiver.

The transmit buffer can only be written when the UDREn flag in the UCSRnA register is set. Data written to UDRn when the UDREn flag is not set, will be ignored by the USART transmitter. When data is written to the transmit buffer, and the transmitter is enabled, the transmitter will load the data into the transmit shift register when the shift register is empty. Then the data will be serially transmitted on the TxDn pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use read-modify-write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

19.10.2 UCSRnA – USART Control and Status Register n A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXCn: USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled, the receive buffer will be flushed and consequently the RXCn bit will become zero. The RXCn flag can be used to generate a receive complete interrupt (see description of the RXCIEn bit).

- **Bit 6 – TXCn: USART Transmit Complete**

This flag bit is set when the entire frame in the transmit shift register has been shifted out and there are no new data currently present in the transmit buffer (UDRn). The TXCn flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXCn flag can generate a transmit complete interrupt (see description of the TXCIEn bit).

- **Bit 5 – UDREn: USART Data Register Empty**

The UDREn flag indicates if the transmit buffer (UDRn) is ready to receive new data. If UDREn is one, the buffer is empty, and therefore ready to be written. The UDREn flag can generate a data register empty interrupt (see description of the UDRIEn bit). UDREn is set after a reset to indicate that the transmitter is ready.

- **Bit 4 – FEn: Frame Error**

This bit is set if the next character in the receive buffer had a frame error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDRn) is read. The FEn bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRnA.

- **Bit 3 – DORn: Data OverRun**

This bit is set if a data overrun condition is detected. A data overrun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive shift register, and a new start bit is detected. This bit is valid until the receive buffer (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

- **Bit 2 – UPEn: USART Parity Error**

This bit is set if the next character in the receive buffer had a parity error when received and the parity checking was enabled at that point (UPMn1 = 1). This bit is valid until the receive buffer (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

- **Bit 1 – U2Xn: Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCMn: Multi-processor Communication Mode**

This bit enables the multi-processor communication mode. When the MPCMn bit is written to one, all the incoming frames received by the USART receiver that do not contain address information will be ignored. The transmitter is unaffected by the MPCMn setting. For more detailed information see Section 19.9 "Multi-processor Communication Mode" on page 158.

19.10.3 UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXCIEn: RX Complete Interrupt Enable n**

Writing this bit to one enables interrupt on the RXCn flag. A USART receive complete interrupt will be generated only if the RXCIEn bit is written to one, the global interrupt flag in SREG is written to one and the RXCn bit in UCSRnA is set.

- **Bit 6 – TXCIEn: TX Complete Interrupt Enable n**

Writing this bit to one enables interrupt on the TXCn flag. A USART transmit complete interrupt will be generated only if the TXCIEn bit is written to one, the global interrupt flag in SREG is written to one and the TXCn bit in UCSRnA is set.

- **Bit 5 – UDRIEn: USART Data Register Empty Interrupt Enable n**

Writing this bit to one enables interrupt on the UDREn flag. A data register empty interrupt will be generated only if the UDRIEn bit is written to one, the global interrupt flag in SREG is written to one and the UDREn bit in UCSRnA is set.

- **Bit 4 – RXENn: Receiver Enable n**

Writing this bit to one enables the USART receiver. The receiver will override normal port operation for the RxDn pin when enabled. Disabling the receiver will flush the receive buffer invalidating the FEn, DORn, and UPEn flags.

- **Bit 3 – TXENn: Transmitter Enable n**

Writing this bit to one enables the USART transmitter. The transmitter will override normal port operation for the TxDn pin when enabled. The disabling of the transmitter (writing TXENn to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the transmit shift register and transmit buffer register do not contain data to be transmitted. When disabled, the transmitter will no longer override the TxDn port.

- **Bit 2 – UCSZn2: Character Size n**

The UCSZn2 bits combined with the UCSZn1:0 bit in UCSRnC sets the number of data bits (character size) in a frame the receiver and transmitter use.

- **Bit 1 – RXB8n: Receive Data Bit 8 n**

RXB8n is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDRn.

- **Bit 0 – TXB8n: Transmit Data Bit 8 n**

TXB8n is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDRn.

19.10.4 UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- **Bits 7:6 – UMSELn1:0 USART Mode Select**

These bits select the mode of operation of the USARTn as shown in Table 19-4.

Table 19-4. UMSELn Bits Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)(1)

Note: 1. See Section 20. "USART in SPI Mode" on page 166 for full description of the master SPI mode (MSPIM) operation

- **Bits 5:4 – UPMn1:0: Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The receiver will generate a parity value for the incoming data and compare it to the UPMn setting. If a mismatch is detected, the UPEn flag in UCSRnA will be set.

Table 19-5. UPMn Bits Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, even parity
1	1	Enabled, odd parity

- **Bit 3 – USBSn: Stop Bit Select**

This bit selects the number of stop bits to be inserted by the transmitter. The receiver ignores this setting.

Table 19-6. USBS Bit Settings

USBSn	Stop Bit(s)
0	1-bit
1	2-bit

- **Bit 2:1 – UCSZn1:0: Character Size**

The UCSZn1:0 bits combined with the UCSZn2 bit in UCSRnB sets the number of data bits (character size) in a frame the receiver and transmitter use.

Table 19-7. UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 – UCPOLn: Clock Polarity**

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOLn bit sets the relationship between data output change and data input sample, and the synchronous clock (XCKn).

Table 19-8. UCPOLn Bit Settings

UCPOLn	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sampled (Input on RxDn Pin)
0	Rising XCKn edge	Falling XCKn edge
1	Falling XCKn edge	Rising XCKn edge

19.10.5 UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

- **Bit 15:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRnH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRnH contains the four most significant bits, and the UBRRnL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the transmitter and receiver will be corrupted if the baud rate is changed. Writing UBRRnL will trigger an immediate update of the baud rate prescaler.

Annexe 2 : Interface TWI/I2C

21.7.1 Master Transmitter Mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 21-11). In order to enter a master mode, a START condition must be transmitted. The format of the following address packet determines whether master transmitter or master receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

TWEN must be set to enable the 2-wire serial interface, TWSTA must be written to one to transmit a START condition and TWINT must be written to one to clear the TWINT flag. The TWI will then test the 2-wire serial bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT flag is set by hardware, and the status code in TWSR will be 0x08 (see Table 21-3). In order to enter MT mode, SLA+W must be transmitted. This is done by writing SLA+W to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

When SLA+W have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in master mode are 0x18, 0x20, or 0x38. The appropriate action to be taken for each of these status codes is detailed in Table 21-3.

When SLA+W has been successfully transmitted, a data packet should be transmitted. This is done by writing the data byte to TWDR. TWDR must only be written when TWINT is high. If not, the access will be discarded, and the write collision bit (TWWC) will be set in the TWCR register. After updating TWDR, the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

This scheme is repeated until the last byte has been sent and the transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	1	X	1	0	X

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

After a repeated START condition (state 0x10) the 2-wire serial interface can access the same slave again, or a new slave without transmitting a STOP condition. Repeated START enables the master to switch between slaves, master transmitter mode and master receiver mode without losing control of the bus.

Table 21-3. Status Codes for Master Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	0	0	1	X	SLA+R will be transmitted; Logic will switch to master receiver mode
0x18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
0x28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
0x30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
0x38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	2-wire Serial Bus will be released and not addressed Slave mode entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free