

MINI-PROJET BSE

COMPLEMENTS TECHNIQUES

Pilotage d'un système de Tri de Colis Pesage et Marquage

Microcontrôleur 8051F020 --Timers, Uarts, Conversion A/N et
Interruptions

1. Environnement et Documentations

Le cœur du système à concevoir est un microcontrôleur. Pour faciliter cette étude, le microcontrôleur, la carte de développement, et l'environnement de développement logiciel utilisés seront les mêmes que ceux étudiés dans les TPs Systèmes à Microprocesseurs.

Il s'agit du microcontrôleur **C8051F020** de la société *Silicon Laboratories*, monté sur une carte de développement **C5051F020DK**. L'outil de développement est l'environnement *Uvision4* de *Keil*, qui permet de programmer aussi bien en C qu'en assembleur.

Pour ce projet, vous allez devoir vous appuyer sur la « **Datasheet complète du 8051F020** » présente sur le E-campus et installée en local sur les machines de TP.

Les différentes phases de conception de ce projet seront traitées séquentiellement durant les séances de TP/TD BSE2, BSE3, BSE4, BSE5 et BSE6.

La séance BSE1 sera consacrée à la découverte du codage en C dans l'environnement Uvision4 et à l'écriture des premiers codes de configuration du microcontrôleur 8051F020.

BSE2 sera consacrée à la découverte du cahier des charges et aux premières réflexions.

BSE3 et BSE4 permettront de d'étudier et de mettre en œuvre tous les périphériques impliqués dans ce mini-projet.

Enfin BSE5 et BSE6 seront consacrées à l'intégration des codes de configuration et d'utilisation des périphériques et au codage de l'application globale.

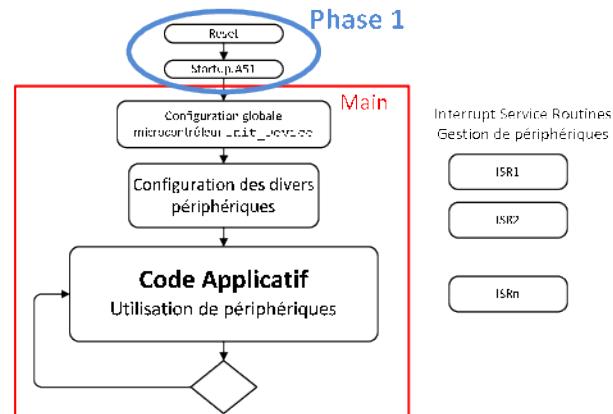


Attention, pensez à conserver vos codes d'une séance à l'autre. Les PCs en salle ne TP ne doivent pas être utilisés pour sauvegarder vos codes d'une séance à l'autre. Et les éventuels rendus E-campus ne pourront pas vous être restitués.

2. Phase 1 - Création du projet sous µVision4.

2.1. Paramètres imposés:

- Microcontrôleur 8051F020 de Silicon laboratories
- Paramétrage par défaut d'un projet sous UV4
- Autoriser l'insertion du code Startup.A51



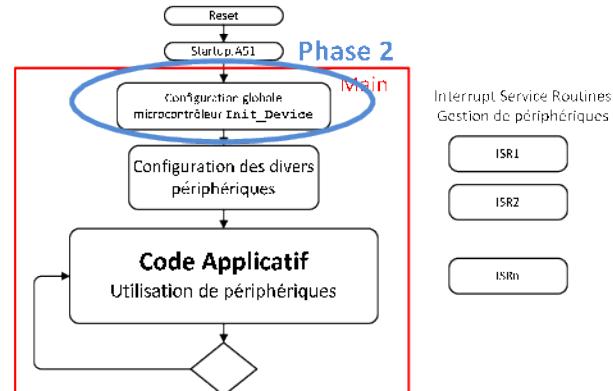
2.2. Contraintes

- Le fichier source contenant entre autre la fonction *Main* sera nommé *BSE_Main.C*.

3. Phase 2 – Mise en place de la configuration globale du microcontrôleur.

3.1. Rappel des configurations à gérer

- Les sources de Reset
- Le Watchdog
- L'horloge
- Les mémoires
- La gestion de la puissance
- L'affectation des broches d'entrée-sortie GPIO / Périphériques
- La configuration des broches réservées aux périphériques



3.2. Paramètres imposés:

- Watchdog désactivé
- SYSCLK = 22,1184 Mhz
- Respecter les affectations des broches I/O du Document « Mini-Projet BSE 2014 » page 10

3.3. Contraintes

- Ce code sera écrit dans un fichier source spécifique : *LIB_BSE_Config_Globale.C* (à créer)
- Créer le fichier *LIB_BSE_Config_Globale.h* associé
- Une seule fonction appelable par le *Main*: *void Init_Device(void)*
- *Init_Device* appellera plusieurs fonctions de configuration (une fonction pour chaque dispositif)

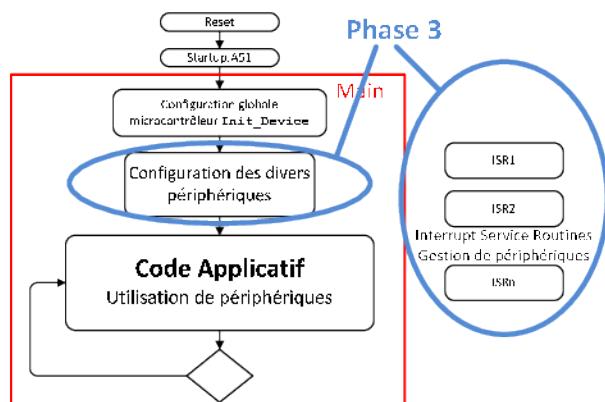
3.4. Suggestions

- Utilisation du **Configuration Wizard 2** pour générer les fonctions de configuration.
- Faire un bilan précis de toutes les broches utilisées dans ce projet par les périphériques du microcontrôleur. Pour chacune de ces broches on s'interrogera sur la configuration de l'étage de sortie : push pull ou drain ouvert.

4. Phase 3 – Configuration et gestion des périphériques

4.1. Inventaire des périphériques à configurer et à gérer

- Les GPIO
- Une interruption externe
- Un Timer
- Une UART
- Un Convertisseur A/N



4.2. Configuration et Gestion des GPIOs

4.2.1. Paramètres imposés:

- Les GPIOs utilisés sont sur les Ports P2 et P3 – Respecter les affectations du cahier des charges. Voir Document « Mini-Projet BSE 2014 » page 10

4.2.2. Contraintes

- Ce code sera écrit dans un fichier source spécifique : *LIB_BSE_GPIO.C* (à créer)
- Créer le fichier *LIB_BSE_GPIO.h* associé
- Code une fonction de configuration : *void Config_GPIO(void)*
- Coder les fonctions de gestion de GPIO :
 - *Void Pulse_P20(void)* : génération d'une impulsion de 500nS sur P2.0
 - *Void Pulse_P21(void)* : génération d'une impulsion de 500nS sur P2.1.
- Proposer un schéma de câblage des diverses GPIO (Boutons Pousoirs, LEDs)
- Coder une fonction *void Demo_GPIO(void)* qui permet de mettre en évidence le bon fonctionnement du code de gestion des GPIO

4.2.3. Suggestions

- Un fichier de déclaration des GPIO est mis à disposition sur le e-campus : *Declarations_GPIO_BSE.h*

4.2.4. Validation



Séance BSE3 - Faire valider la création de projet, les fonctions de configuration globale et les fonctions de configuration et de gestion des GPIOs via un programme de démonstration.

4.3. Configuration et Gestion de l'interruption externe INT1

4.3.1. Paramètres imposés:

- Utilisation impérative de l'interruption externe INT1

4.3.2. Contraintes

- Ce code sera écrit dans un fichier source spécifique : *LIB_BSE_INT_Ext.C* (à créer)
- Créer le fichier *LIB_BSE_INT_Ext.h* associé
- Code une fonction de configuration : *void Config_INT1(void)*

- Coder le squelette de la fonction d'interruption INT1 *ISR_INT1*. On utilisera le port P3.4 (Signal Test4) pour visualiser sur oscilloscope le déroulement de l'interruption (Test4 est forcé à 1 au début de la fonction d'interruption, puis est remis à zéro à la fin de cette fonction).
- Faites une démonstration du code réalisé.

4.3.3. Validation



Séance BSE3 - Faire valider la mise en place de l'INT1 via un programme de démonstration.

4.4. Configuration et Gestion de du Timer2

4.4.1. Paramètres imposés:

- Utilisation impérative du Timer 2 qui fonctionnera en base de temps.
- La fréquence de fonctionnement du Timer2 sera fixée à 200Hz (compte tenu des contraintes du cahier des charges).

4.4.2. Contraintes

- Ce code sera écrit dans un fichier source spécifique : *LIB_BSE_Timers.C* (à créer)
- Créer le fichier *LIB_BSE_Timers.h* associé
- Code une fonction de configuration : *void Config_Timer2 (void)*
- Coder le squelette de la fonction d'interruption *ISR_Timer2*. On utilisera le port P3.5 (Signal Test5) pour visualiser sur oscilloscope le déroulement de l'interruption (Test5 est forcé à 1 au début de la fonction d'interruption, puis est remis à zéro à la fin de cette fonction).
- Le code d'interruption intégrera la gestion d'une horloge temps réel. On utilisera les variables globales suivantes :
 - *unsigned char RTC_5ms*
 - *unsigned char RTC_Secondes*
 - *unsigned char RTC_Minutes*
 - *unsigned char RTC_Heures*
- Prévoir une fonction de remise à zéro de l'horloge temps réel *void Clear_RTC (void)*.
- Coder une démonstration du code réalisé.

4.4.3. Validation



Séance BSE3 - Faire valider la mise en œuvre du Timer 2 et de l'horloge temps réel au travers d'un code de démonstration.

4.5. Configuration et Gestion de l'UART0

4.5.1. Paramètres imposés:

- Utilisation impérative de l'UART0 avec le Timer 1 comme source d'horloge.
- Les paramètres de la liaison série sont : 115200 Baud, 8 bits de données, 1 Stop bit, pas de parité.

4.5.2. Contraintes

- Ce code sera écrit dans un fichier source spécifique : *LIB_BSE_UART.C* (à créer)
- Créer le fichier *LIB_BSE_UART.h* associé
- Coder une fonction de configuration : *void CFG_Clock_UART0 (void)* Cette fonction configure le timer 1 comme source d'horloge pour l'UART0
- Coder une fonction de configuration : *void CFG_UART0 (void)*
- Coder une fonction de gestion de l'UART : *char Putchar(char c, charg csg_tempo)* – Transmission à l'UART d'un caractère à transmettre – **Voir Algorithme en Annexe**
- Coder une fonction de gestion de l'UART : *char Send_String(char *char_ptr)* qui se charge de transmettre sur la liaison série la chaîne de caractère pointée par *char_ptr*, et ce, à l'aide de la fonction *Putchar*. Pour respecter les conventions du langage C, le caractère NUL (0X00) sera le caractère de fin de chaîne. Cette fonction renverra le nombre de caractères transmis ou la valeur 0 en cas d'erreur.
- Coder une fonction de gestion de l'UART : *char Getchar(void)* qui va lire le dernier caractère reçu sur la liaison série – **Voir Algorithme en Annexe**
- Faites une démonstration du code réalisé. Un programme d'écho de caractères fera très bien l'affaire. Par le PC on enverra un caractère, le 8051F020 le recevra grâce à *getchar* et le retournera en écho au PC avec *Putchar*. Puis le 8051 enverra une chaîne de caractères avec *Send_string*.

4.5.3. Validation



Séance BSE4 - Faire valider la mise en œuvre de l'UART0 et des fonctions demandées, au travers d'un code de démonstration.

4.6. Configuration et Gestion de l'ADC0

4.6.1. Paramètres imposés:

- Utilisation impérative de l'ADC0 comme convertisseur Analogique Numérique.
- Conversion en unipolaire de la voie AIN0.3.
- On ne s'intéresse qu'aux 8 bits de poids fort, du résultat de la conversion.
- La conversion sera déclenchée par une mise à 1 de AD0BUSY.
- La fin de conversion sera détectée par scrutation du drapeau AD0INT.

4.6.2. Contraintes

- Ce code sera écrit dans un fichier source spécifique : *LIB_BSE_ADC.C* (à créer).
- Créer le fichier *LIB_BSE_ADC.h* associé
- Coder une fonction de configuration : *void CFG_VREF (void)* Cette fonction active la source de tension de référence du 8051F020.
- Coder une fonction de configuration : *void CFG_ADC0 (void)* Cette fonction configure le convertisseur ADC0 pour faire des conversions sur AIN0.3. Le déclenchement de la conversion sera fait logiciellement.
- Coder une fonction de gestion de l'ADC: *unsigned char ACQ_ADC (void)* Cette fonction déclenche une conversion sur AIN0.3, attend la fin de conversion, et retourne le résultat.
- Faites une démonstration du code réalisé. Nous mettrons à disposition sur le e-campus un code pour piloter un convertisseur Numérique Analogique (DAC). Il sera donc possible de coder une démonstration, qui fait une conversion (dans l'interruption timer2 par exemple) et qui renvoie le résultat sur un DAC.

4.6.3. Validation

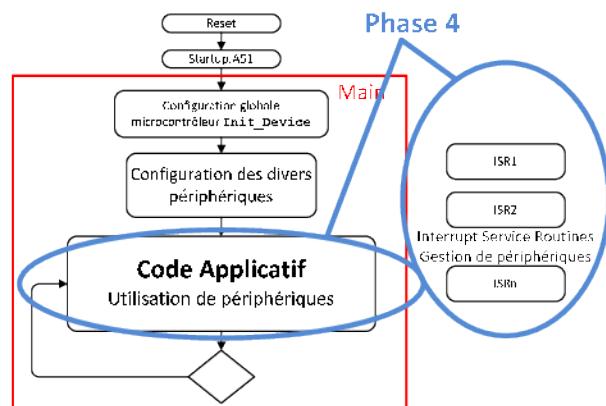


Séance BSE4 - Faire valider la mise en œuvre de l'ADC0 et des fonctions demandées, au travers d'un code de démonstration.

5. Phase – Code Application

5.1. Paramètres imposés:

- Réutiliser les codes de gestion des divers périphériques.



5.2. Contraintes

- Coder une fonction de gestion *void CONV_Pes_Val(unsigned char Pes_value, char *String_Pes_Val)* pour transformer la valeur binaire Pes_value (typiquement le résultat produit par la fonction ACG_ADC en une chaîne de caractères ASCII exprimant le poids du colis en grammes.

5.3. Suggestions

- Procéder avec méthode en implémentant les actions les unes après les autres, en ayant pris le soin de réfléchir auparavant à l'architecture globale du logiciel applicatif.

5.3.1. Validation



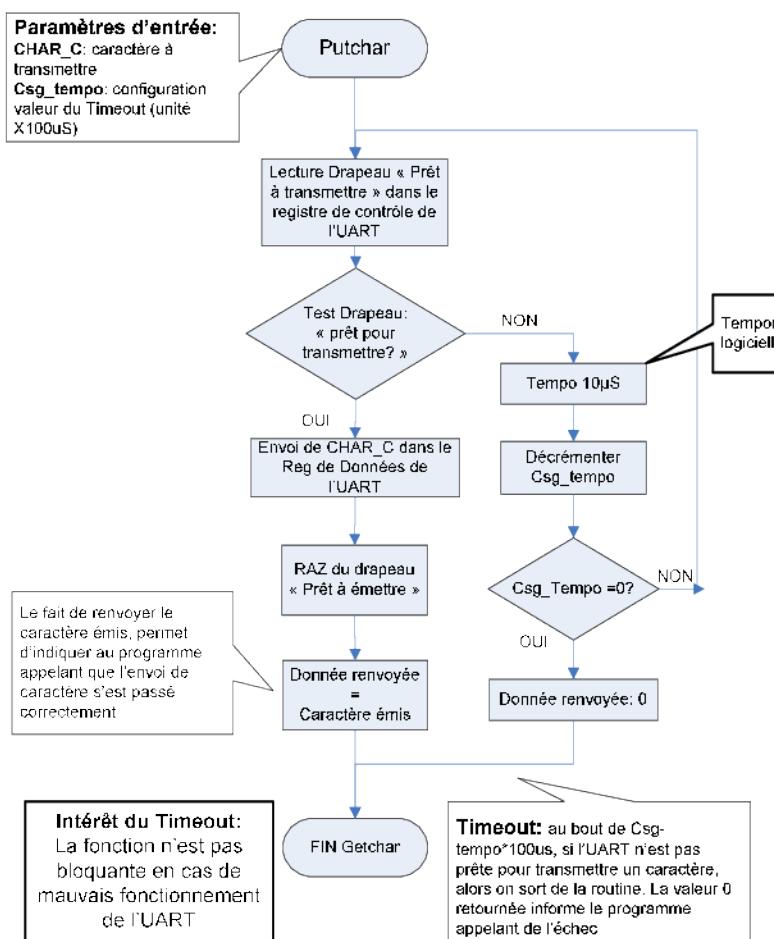
Séance BSE5 et BSE6 - Faire valider vos résultats finaux.

6. ANNEXES

6.1. ALGORIGRAMMES

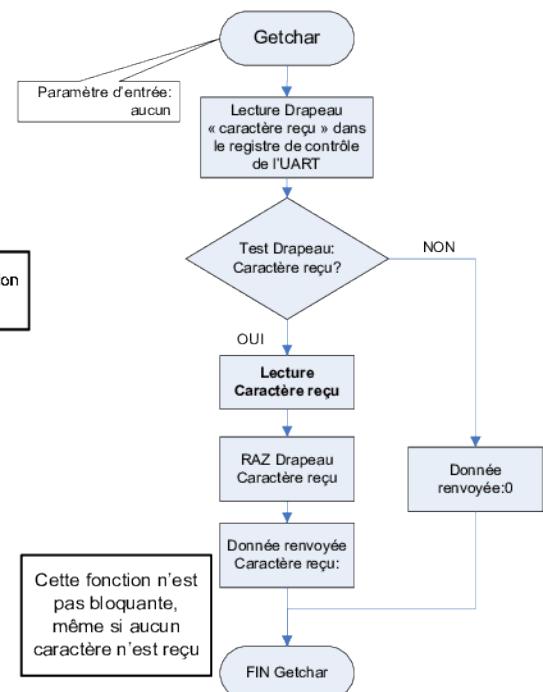
Algorigramme PUTCHAR sur 8051

Prototype: Char PUTCHAR (char c, unsigned char csg_tempo)



Algorigramme GETCHAR sur 8051

Prototype: Char GETCHAR ()

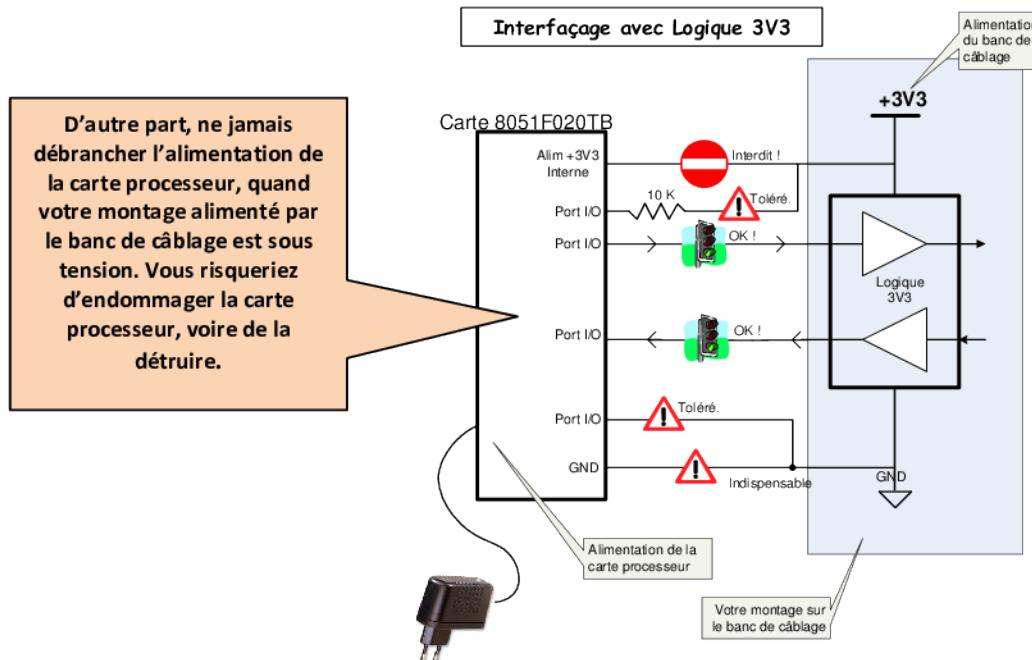


6.2. Conseils de mise en œuvre.

Attention : la carte d'évaluation fonctionne avec une alimentation interne de +3V3 et génère des signaux en logique CMOS +3.3V. Si vous devez ajouter des fonctions logiques, utilisez exclusivement des circuits de la famille 74HC alimentés en 3V3.

Pour pallier ce risque, respectez strictement l'ordre de mise en route et d'arrêt des alimentations :

- Pour la mise en route, alimenter la carte processeur, puis alimenter votre montage avec l'alimentation du banc de câblage. Pour plus de sécurité, il est préférable d'éviter d'utiliser l'interrupteur général du banc de câblage, et d'avoir plutôt recours à un fil que l'on branche ou débranche. En effet à chaque sollicitation de l'interrupteur général, de fortes variations de tension sont constatées sur les alimentations, variations de tension pouvant endommager les circuits alimentés en basse tension.
- Pour l'arrêt, couper l'alimentation de votre montage (par la méthode préconisée si dessus), puis l'alimentation de la carte processeur.



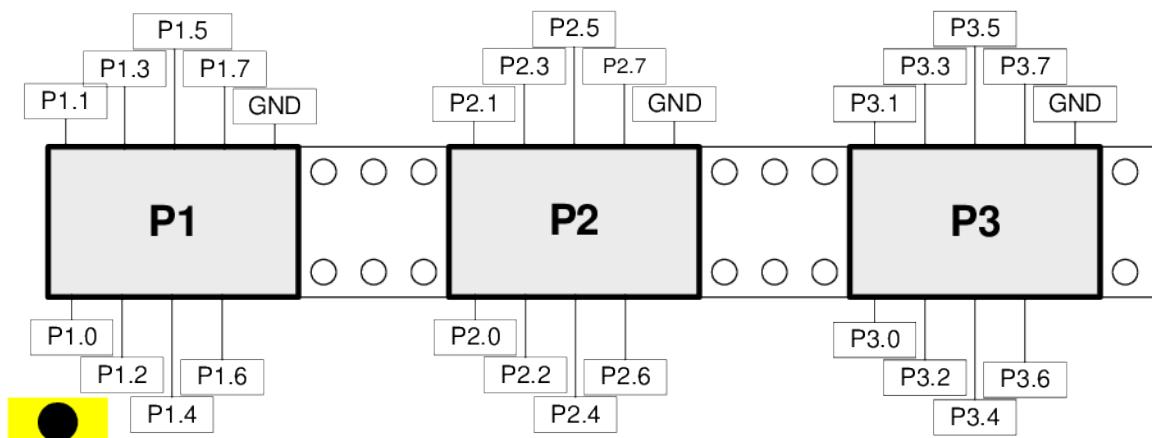
6.3. Brochage du Connecteur d'extension du Bus micropuceur.

Attention au sens de branchement du connecteur d'extension du bus 8051.

Un repère a été ajouté sur ce connecteur pour vous faciliter la tâche.

Ce connecteur est fragile, manipulez-le avec soin. Pour l'ôter du banc de câblage, à la fin du TP, utilisez de préférence un tournevis pour l'extraire sans tordre les pattes de connexion.

Connecteur Ports I/O P1-P2-P3



P1, P2, et P3 Connecteurs d'extension de la carte SILABS

Repère

La broche 1 de P0 est repérée par le marquage suivant :



6.4. Brochage du Connecteur DIN41612 de la carte 8051F020

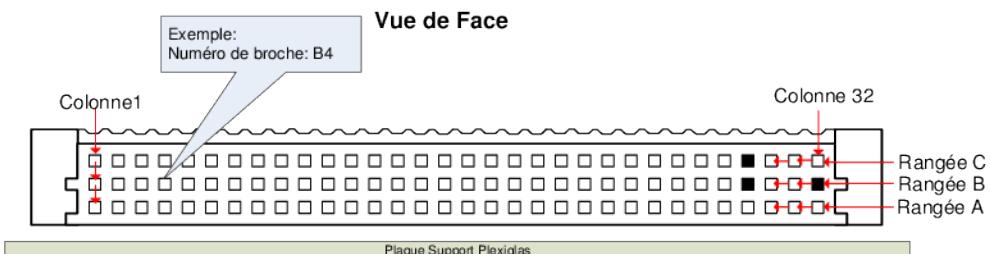
Pin #	Description
A-1	+3 VD2 (+3.3 VDC)
A-2	MONEN
A-3	P1.5
A-4	P1.2
A-5	P2.7
A-6	P2.4
A-7	P2.1
A-8	P3.6
A-9	P3.3
A-10	P3.0
A-11	P0.5
A-12	P0.2
A-13	P7.7
A-14	P7.4
A-15	P7.1
A-16	P6.6
A-17	P6.3
A-18	P6.0
A-19	P5.5
A-20	P5.2
A-21	P4.7
A-22	P4.4
A-23	P4.1
A-24	TCK
A-25	/RST
A-26	AGND (Analog Gnd)
A-27	CP1-
A-28	CP0+
A-29	VREF0
A-30	AIN0.6
A-31	AIN0.3
A-32	AIN0.0

Pin #	Description
B-1	DGND (Digital Gnd)
B-2	P1.7
B-3	P1.4
B-4	P1.1
B-5	P2.6
B-6	P2.3
B-7	P2.0
B-8	P3.5
B-9	P3.2
B-10	P0.7
B-11	P0.4
B-12	P0.1
B-13	P7.6
B-14	P7.3
B-15	P7.0
B-16	P6.5
B-17	P6.2
B-18	P5.7
B-19	P5.4
B-20	P5.1
B-21	P4.6
B-22	P4.3
B-23	P4.0
B-24	TDI
B-25	DGND (Digital Gnd)
B-26	DAC1
B-27	CP1+
B-28	VREF
B-29	VREF1
B-30	AIN0.5
B-31	AIN0.2
B-32	AGND (Analog Gnd)

Pin #	Description
C-1	XTAL1
C-2	P1.6
C-3	P1.3
C-4	P1.0
C-5	P2.5
C-6	P2.2
C-7	P3.7
C-8	P3.4
C-9	P3.1
C-10	P0.6
C-11	P0.3
C-12	P0.0
C-13	P7.5
C-14	P7.2
C-15	P6.7
C-16	P6.4
C-17	P6.1
C-18	P5.6
C-19	P5.3
C-20	P5.0
C-21	P4.5
C-22	P4.2
C-23	TMS
C-24	TDO
C-25	VUNREG
C-26	DAC0
C-27	CP0-
C-28	VREFD
C-29	AIN0.7
C-30	AIN0.4
C-31	AIN0.1
C-32	AV+ (+3.3 VDC Analog)

Table 6. J24 Pin Descriptions

Brochage du connecteur D41612 96 Cts de la carte C8051FX20-TB



6.5. Table ASCII

ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	44	2C	,	88	58	X
1	1	SOH	45	2D	-	89	59	Y
2	2	STX	46	2E	.	90	5A	Z
3	3	ETX	47	2F	/	91	5B	\
4	4	EOT	48	30	0	92	5C	\`
5	5	ENQ	49	31	1	93	5D	_
6	6	ACK	50	32	2	94	5E	^
7	7	BEL	51	33	3	95	5F	
8	8	BS	52	34	4	96	60	~
9	9	HT	53	35	5	97	61	a
10	A	LF	54	36	6	98	62	b
11	B	VT	55	37	7	99	63	c
12	C	FF	56	38	8	100	64	d
13	D	CR	57	39	9	101	65	e
14	E	SO	58	3A	:	102	66	f
15	F	SI	59	3B	;	103	67	g
16	10	DLE	60	3C	<	104	68	h
17	11	DC1	61	3D	=	105	69	i
18	12	DC2	62	3E	>	106	6A	j
19	13	DC3	63	3F	?	107	6B	k
20	14	DC4	64	40	@	108	6C	l
21	15	NAK	65	41	A	109	6D	m
22	16	SYN	66	42	B	110	6E	n
23	17	E1B	67	43	C	111	6F	o
24	18	CAN	68	44	D	112	70	p
25	19	EM	69	45	E	113	71	q
26	1A	SUB	70	46	F	114	72	r
27	1B	ESC	71	47	G	115	73	s
28	1C	FS	72	48	H	116	74	t
29	1D	GS	73	49	I	117	75	u
30	1E	RS	74	4A	J	118	76	v
31	1F	US	75	4B	K	119	77	w
32	20	SP	76	4C	L	120	78	x
33	21	!	77	4D	M	121	79	y
34	22	"	78	4E	N	122	7A	z
35	23	#	79	4F	O	123	7B	{
36	24	\$	80	50	P	124	7C	
37	25	%	81	51	Q	125	7D	}
38	26	&	82	52	R	126	7E	~
39	27	'	83	53	S	127	7F	
40	28	(84	54	T			
41	29)	85	55	U			
42	2A	*	86	56	V			
43	2B	+	87	57	W			

6.6. Carte C8051F020DK

The C8051F02x Development Kit includes a target board with a C8051F020 device pre-installed for evaluation and preliminary software development. Numerous input/output (I/O) connections are provided to facilitate prototyping using the target board. Refer to Figure 2 for the locations of the various I/O connectors.

P1	Power connector	(accepts input from 7 to 15 VDC unregulated power adapter)
J1	Connects SW2 to P3.7 pin	
J3	Connects LED D3 to P1.6 pin	
J4	JTAG connector for Serial Adapter interface	
J5	DB-9 connector for UART0 RS232 interface	
J6	Jumper to connect UART0 TX (P0.0) to DB9	
J9	Jumper to connect UART0 RX (P0.1) to DB9	
J11	Analog loopback connector	
J12-J19	Port 0 - 7 connectors	
J20	Analog I/O terminal block	
J22	VREF connector	
J23	VDD Monitor Disable	
J24	96-pin Expansion I/O connector	

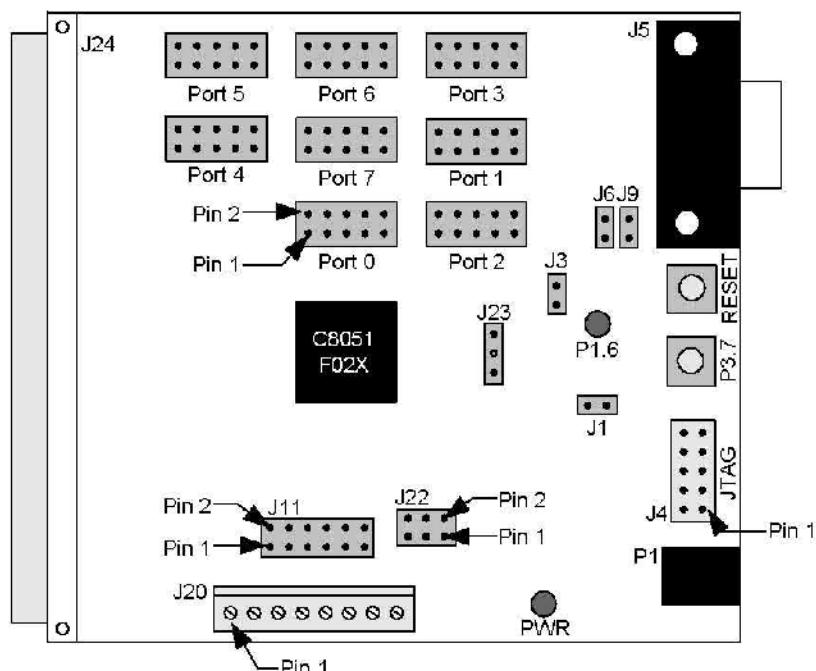
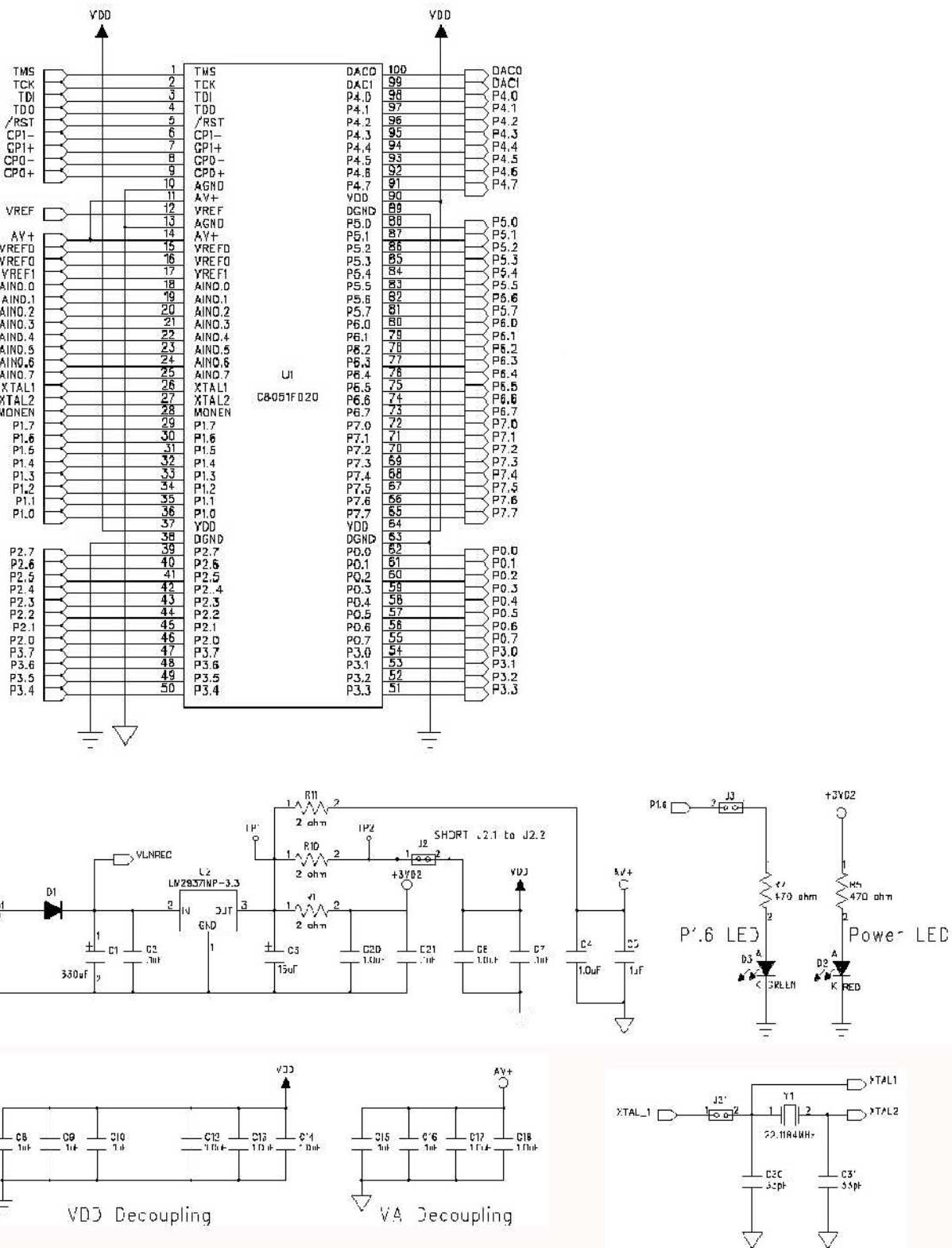
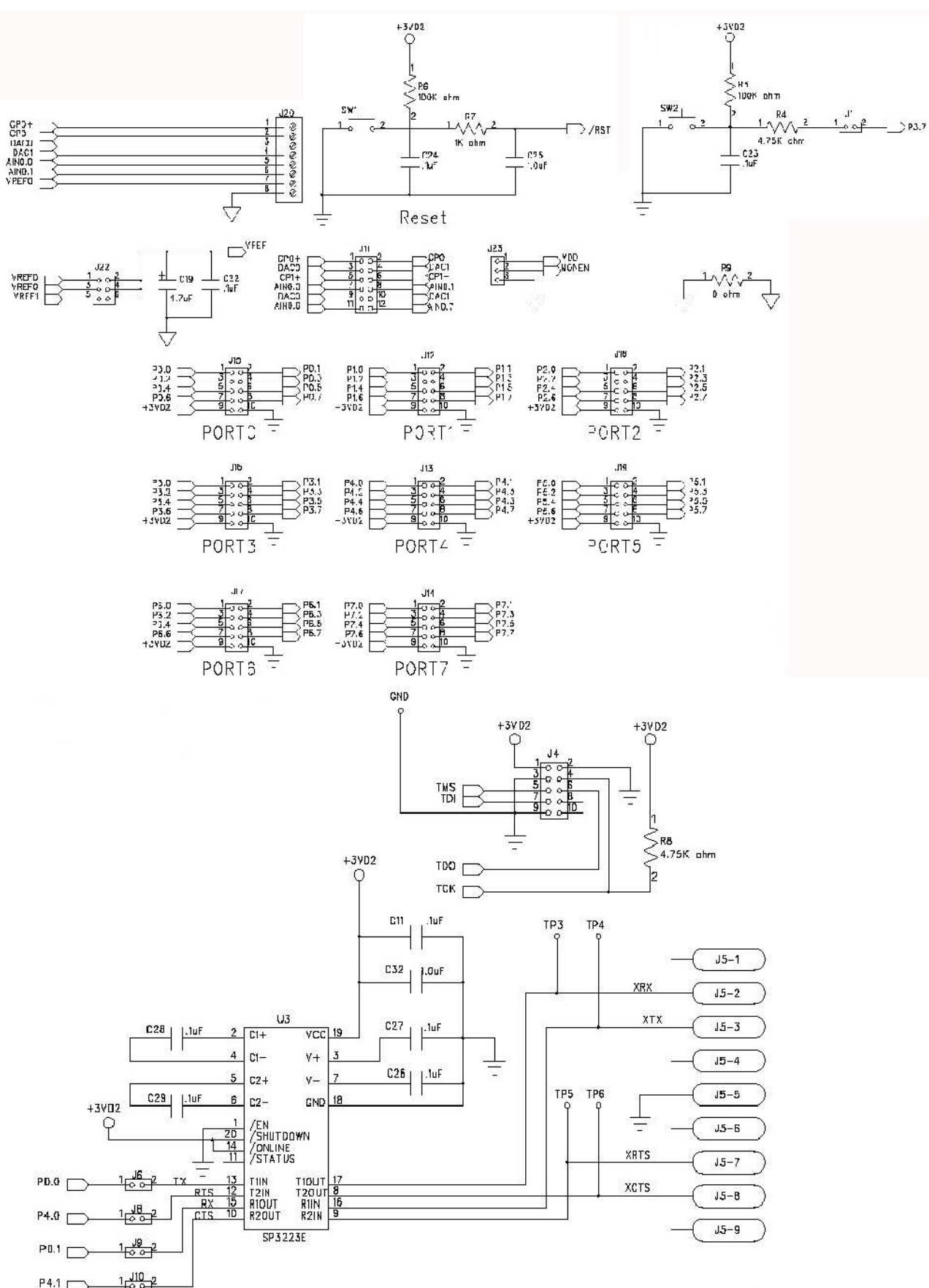


Figure 2. C8051F020 Target Board

6.7. Schéma de la carte C8051FX20-TB





6.8. Spécifications des Diodes LED à disposition

Round LED Bright Red, 5mm


Specifications:

Dice material	: GaP
Emitted colour	: Red
Lens colour	: Red Diffused
Peak wavelength	: 697nm
Viewing angle	: 45°
Luminous intensity (IV)	: 2.8mcd


Absolute Maximum Ratings ($T_a = 25^\circ\text{C}$)

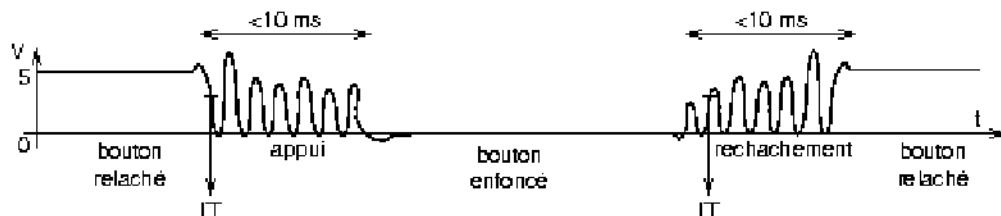
Reverse Voltage	5V
Reverse Current	10µA ($V_R = 5\text{V}$)
Operating Temperature Range	-40°C to +85°C
Storage Temperature Range	-40°C to +100°C
Lead Soldering Temperature Range 1.6mm (1/16 inch) from body	260°C for 5 Seconds

Electrical/Optical Characteristics at $T_a = 25^\circ\text{C}$

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Test
Luminous Intensity	IV	1.3	2.8	4.2	mcd	
Viewing Angle	$2\theta_{1/2}$	-	45	-	degrees	IF = 20mA
Peak Emission Wavelength	λ_P	-	697	-	nm	-
Dominant Wavelength	λ_D	-	650	-	nm	-
Spectral Line Half-Width	$\Delta\lambda$	-	90	-		-
Forward Voltage	V_F	1.7	2.1	2.6	V	IF = 20mA
Power Dissipation	Pd	-	-	45	-	-
Peak Forward Current (Duty 1/10 at 1KHz)	IF (Peak)	-	-	50	-	-
Recommended Operating Current	IF (Rec)	-	10	-	mA	-

6.9. Spécification des boutons poussoirs

Attention à la problématique des rebonds !!



Source : <https://www-asim.lip6.fr/trac/seesi-pmcii/wiki/MicroTmeInter>



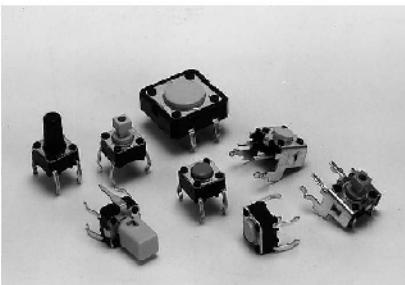
[◀ Back](#)

Mechanical Key Switch

B3F

A Wide Range of Models: 6 x 6 mm, 12 x 12 mm, Vertical and High-force.

- Excellent tactile feel combined with high life expectancy.
- Available in stick packaging for automatic mounting to PCBs.
- Up to 2.55 N (260 gf) operating force for a firmer operating touch.
- Pushbuttons available in nine colors.



Plunger type	Flat	Projected	Extended	Flat (vertical)	Projected (vertical)
Appearance					

Specifications

■ Ratings

Switching capacity	5 to 24 VDC, 1 to 50 mA (resistive load)
Insulation voltage	30 VDC

■ Characteristics

Contact form	SPST-NO
Contact resistance	100 mΩ max. (at 5 VDC, 1 mA; initial value.)
Insulation resistance	100 MΩ min. (at 250 VDC)
Dielectric strength	500 VAC, 50/60 Hz for 1 min
Bounce time	5 ms max.
Vibration resistance	Malfuction: 10 to 60 Hz, 1.5 mm double amplitude
Shock resistance	Destruction: 1,000 m/s ² min. (Approx. 100 G min.) Malfuction: 100 m/s ² min. (Approx. 10G min.)
Life expectancy	B3F-1□□□, 3□□□ 100-gf OF: 1,000,000 operations min. 150-gf OF: 300,000 operations min. 200-gf OF: 100,000 operations min. B3F-4□□□ 130-gf OF: 3,000,000 operations min. 260 gf OF: 1,000,000 operations min. B3F-5□□□: 10,000,000 operations min. B3F-9□□□: 1,000,000 operations min.
Ambient temperature	25°C to 70°C (will no icing)
Ambient humidity	35% to 85%
Weight	6 x 6 mm models: approx. 0.26 g 12 x 12 mm models: approx. 0.86 g

■ Operating Characteristics

Model	B3F-1□□□, -3□□□			B3F-4□□□, -6□□□, -9□□□	
	100 gf models	150 gf models	260 gf models	130 gf models	260 gf models
B3F-1□□□, -3□□□	B3F-1□□□, -3□□□	B3F-1□□□, -3□□□	B3F-1□□□, -3□□□	B3F-4□□□, -6□□□, -9□□□	B3F-4□□□, -6□□□, -9□□□
Operating force (OF)	0.90±0.32 N (100–30 gf)	1.47±0.49 N (150–50 gf)	2.55±0.69 N (260–70 gf)	1.27±0.49 N (130–60 gf)	2.55±0.69 N (260–70 gf)
Recess force (RF min.)	0.2 N (20 gf)	0.49 N (50 gf)	0.49 N (50 gf)	0.29 N (30 gf)	0.49 N (50 gf)
Pretravel (PT)	0.25±0.2/-0.1 mm			0.3±0.2/-0 mm	

6.10. Check-list - Règles de Codage dans les systèmes embarqués
CHECK-LIST – Commentaires d'entête de fichier source

NB: Selon le contenu des fichiers source, certaines rubriques peuvent être sans objet.

ID	Commentaires Globaux en entête de chaque fichier source	Pass	Fail
1.1	Nom du fichier explicite		
1.2	Copyright		
1.3	Auteurs		
1.4	Date de dernière modification		
1.5	Version		
1.6	Objectifs du code		
1.7	Dépendances Matérielles « cible »		
1.8	Dépendances Matérielles « extérieures» - Broches I/O utilisées		
1.9	Dépendances de type communication/protocolaires		
1.10	Explications sur le fonctionnement général du code		
1.11	Choix Technologiques		
1.12	Tests réalisés – Code validé ?		
1.13	Chaîne de compilation		
1.14	Liens vers les documents en lien ou utiles au projet		
1.15	Commentaires sur les variables globales et les constantes		

CHECKLIST – Commentaires d'entête de fonction

NB: Selon le contenu des fichiers source, certaines rubriques peuvent être sans objet.

ID	Commentaires d'entête de Fonctions	Pass	Fail
2.1	Choix explicite des noms de fonctions		
2.2	Objectifs de la fonction		
2.3	Expliciter le sens et la forme des arguments		
2.4	Variables globales manipulées non passés en argument		
2.5	Description et type de la valeur renvoyée		
2.6	Principe de fonctionnement		
2.7	Pré et Post conditions		
2.8	Tests effectués		

ID	Commentaires supplémentaires relatifs aux fonctions de configuration et d'utilisation de périphériques	Pass	Fail
3.1	Expliquer le fonctionnement du périphérique (modes....)		
3.2	Enumérer tous les registres modifiés		
3.3	Expliciter les dépendances (horloge interne, autres configurations ...)		
3.4	Broches I/O utilisées		
3.5	Citer les pages de datasheet utilisées		

CHECKLIST – Règles de codage

ID		Pass	Fail
4.1	Ne pas manipuler des registres de périphériques en dehors des fonctions de configuration et d'utilisation de périphérique.		
4.2	S'imposer l'usage de masques pour éviter la manipulation de bits non désirés de registres dans les fonctions de gestion de périphérique		
4.3			
4.4			
4.5			

6.11. Méthodologie de développement matériel et logiciel d'un petit système embarqué

Type de l'approche :

- **Matériel** : 1 seul microcontrôleur
- **Logiciel** : codage en C « from scratch » - Pas de librairie de périphériques, pas de noyau temps réel
- Conception à partir d'un cahier des charges décrivant l'architecture matérielle

NB : cette méthode ne prétend pas s'appliquer pour la conception de systèmes embarqués complexes, par contre, elle s'applique parfaitement au mini-projet BSE.

CHECK-LIST – Conception matérielle (ordre chronologique)

ID		Pass	Fail
M_1.1	Identifier tous les éléments du dispositif		
M_1.2	Représenter toutes les connexions entre les éléments du dispositif et préciser leur type - Réalisation d'un schéma bloc		
M_1.3	Vérifier la compatibilité électrique entre les connexions des divers éléments (Courants, Tensions, fréquences....)		
M_1.4	Réaliser un schéma électrique complet		
M_1.5	Vérifier le câblage réalisé, avant de mettre sous tension		
M_1.6	Définir les consignes de séquencement des alimentations pour la mise en route du dispositif		
M_1.7			
M_1.8			
M_1.9			
M_1.10			

CHECK-LIST – Conception logicielle (ordre chronologique)

ID		Pass	Fail
M_2.1	Spécifier l'architecture logicielle globale de l'application		
M_2.2	Inventorier tous les périphériques à mettre en œuvre		
M_2.3	Inventorier pour chaque périphérique les fonctions à coder en indiquant quelles sont les fonctions de configuration et les fonctions d'utilisation du périphérique (les fonctions d'interruption en font partie)		
M_2.4	Spécifier les paramètres de configurations globales du microcontrôleur (Horloge, Ports I/O etc....)		
M_2.5	Enumérer les fonctions applicatives. Ces fonctions apportent une réponse au cahier des charges. Elles s'appuient entre autre, sur les fonctions d'utilisation des périphériques.		
M_2.6	Dresser un tableau récapitulatif qui fait apparaître l'ensemble des fonctions appelées dans le Main et dans les divers programmes d'interruption et ce, directement ou indirectement		
M_2.7	Codage et test unitaire des fonctions de configuration globale L'ensemble de ces fonctions sera contenu dans un seul fichier source		
M_2.8	Codage et test unitaire des fonctions de configuration et d'utilisation des divers périphériques. Un fichier source pour chaque périphérique utilisé, contenant les fonctions de configuration et d'utilisation de ce périphérique.		
M_2.9	Codage et test unitaire de toutes les fonctions applicatives.		
M_2.10	Imaginer un plan d'intégration globale en spécifiant plusieurs scénarios de test permettant d'obtenir une couverture totale de code.		