

CardS12

Hardware-
Version 1.00

Manuel utilisateur

14 février 2004

Copyright (C)2003 par MCT Elektronikladen GbR
Hohe Str. 9-13, D-04107 Leipzig
Tél: +49-(0)341-2118354 Fax: +49-(0)341-2118355
Email: mct@elektronikladen.de
Web: <http://www.elektronikladen.de/mct>

Le manuel et le produit décrit dans ce document ont été conçus avec la plus grande attention par le fabricant. Tous les efforts ont été mis en œuvre pour éviter les anomalies. Toutefois, nous ne pouvons garantir que ce dernier soit à 100% exempt de toute erreur.

L'entière responsabilité du fabricant vis-à-vis de votre recours se limitera exclusivement selon le choix du fabricant au remboursement de la carte au prix payé ou à sa réparation ou à son échange. Le fabricant dément toutes autres garanties, exprimées ou implicites, se référant mais sans limitation aux garanties implicites de la valeur marchande du produit (incluant les documents écrits associés, la partie matériel et les logiciels).

En aucun cas le fabricant ou ses distributeurs ne pourront être tenus responsables de dommages quels qu'ils soient (intégrant, mais sans limitation, les dommages pour perte de bénéfice commercial, interruption d'exploitation commerciale, perte d'informations et de données à caractère commercial ou de toute autre perte financière) provenant de l'utilisation ou de l'incapacité d'utiliser le produit, même si le constructeur a été informé de la possibilité de tels dommages. Le produit n'est pas conçu, destiné ou autorisé pour l'usage d'applications dans lesquelles une défaillance du produit pourrait créer une situation dangereuse pouvant entraîner des blessures ou la mort de personnes. Si vous utilisez le produit volontairement ou involontairement pour de telles applications non autorisées, vous vous engagez à soustraire le fabricant et ses distributeurs de toute responsabilité et de toute demande de dédommagement, même si le fabricant a été négligent en ce qui concerne la conception et la mise en œuvre du produit.

Les caractéristiques du produit et les prix peuvent changer sans aucun avertissement préalable.

Toutes les marques déposées sont la propriété de leurs détenteurs respectifs.

Sommaire

1. Vue d'ensemble	4
Données techniques	4
Contenu du starter-kit	5
2. Mise en oeuvre rapide	6
3. Schéma de repérage des composants	7
4. Cavaliers et ponts de soudures	8
Cavaliers	8
Ponts de soudures	8
5. Dimensions mécaniques	10
6. Description du circuit	11
Schéma théorique	11
Cœur du contrôleur - Alimentation	11
Génération du Reset	12
Génération horloge et PLL	13
Modes opératoires - Support BDM	15
Convertisseur A/N intégré	16
Mémoire EEPROM intégrée	17
Témoin LED	19
Interface RS-232	19
Bus SPI	21
Bus IIC	22
Interface CAN	22
7. Conseils pour vos applications	23
Comportement après Reset	23
Code de démarrage	23
Informations additionnelles sur le Web	23
8. Moniteur TwinPEEKs	24
Communication série	24
Fonction Autostart	24
Accès en écriture à la Flash et l'EEPROM	24
Ré-acheminement des vecteurs d'interruption	25
Utilisation	28
Commandes du Moniteur	28
9. Organisation mémoire	32

1. Vue d'ensemble

Simple d'exploitation et économique, la CardS12 est une platine microcontrôlée de la taille d'une carte de crédit conçue autour d'un microcontrôleur 16 bits Motorola™ de la famille HCS12. Pouvant faire office de système d'évaluation ou d'outil de développement pour le prototypage rapide, elle s'apparente également à une solution universelle pour la réalisation d'applications en petites et moyennes séries.

La CardS12 est équipée d'un microcontrôleur MC9S12D64. Ce dernier contient un "cœur" CPU 16 bits HCS12, 64 KB de mémoire Flash, 4 KB de RAM, 1 KB d'EEPROM et une grande quantité de blocs/fonctions périphériques tels que SCI, SPI, CAN, IIC, Timer, PWM, ADC et des "E/S" pour usage général. Le MC9S12D64 dispose d'une gestion totale des données en mode 16 bits. Un circuit PLL intégré permet de moduler les performances de la carte en fonction de sa consommation selon les spécificités de l'application envisagée.

A noter qu'il existe une très large gamme de logiciels de développement pour les microcontrôleurs HCS12 (moniteurs, compilateurs "C", débogueur "BDM") destinés d'accélérer vos phases de développements.

Données techniques

- MCU MC9S12D64 en boîtier LQFP112 (CMS)
- CPU HCS12 16 bits, utilisant le même modèle de programmation et le même jeu de commandes que les HC12
- Quartz d'horloge 16 MHz, horloge bus jusqu'à 25 MHz via PLL
- Mémoire: 64 KB Flash, 1 KB EEPROM, 4 KB RAM
- 2 x SCI – interface asynchrone série (ex: RS232, LIN)
- 1 x SPI - interface asynchrone série
- 1 x IIC – Bus Inter-IC
- 1 x msCAN-Module (compatible CAN 2.0A/B) - Un canal est doté d'un driver d'interface physique haute vitesse

- 8 x Timer 16 Bits (Input Capture/Output Compare)
- 8 x PWM (Pulse Width Modulator)
- 16-canaux de conversion A/N (résolution 10 bits)
- Connecteur 6 pts standard pour interface BDM (Background Debug Mode)
- Circuit LVI spécial (contrôleur de reset)
- Deux interfaces séries avec transceiver RS232 (ex: pour connexion à un PC)
- Le deuxième port série peut être utilisé pour piloter directement un afficheur LCD à commande série
- Led de visualisation
- Bouton "RESET"
- Jusqu'à 87 "E/S" libres pour usage général
- Tous les signaux d'E/S sont repris sur des connecteurs
- Alimentation: 5 V – Consommation: 50 mA typ.
- Dimensions de la carte: 2.1" x 3.4"

Contenu du starter-kit

- Carte contrôleur avec MC9S12D64
- Moniteur TwinPEEKs (dans la mémoire Flash du MCU)
- Câble RS232 (Sub-D9)
- Deux connecteurs (2 x 25 broches), connecteur alimentation
- Manuel utilisateur (ce manuel)
- Schéma théorique de la carte
- CD-ROM: contenant un logiciel assembleur, des data-sheets, le manuel de référence des CPU12, des exemples de codes, un compilateur "C" (version d'évaluation), etc.

2. Mise en œuvre rapide

C'est parce que personne n'aime lire les "gros" manuels que nous avons résumé dans ce chapitre les points les plus importants à savoir. Si vous désirez des informations additionnelles, reportez-vous s'il vous plait aux chapitres plus détaillés de ce manuel.

Voici comment vous pouvez démarrer:

- Vérifiez l'état de la carte et l'absence de dommage potentiellement subit lors du transport du paquet.
- Connectez le contrôleur au port série RS232 d'un PC. La connexion entre la CardS12 (interface SER0, connecteur X3) et le PC sera facilement réalisée en utilisant le câble plat livré avec la carte.
- Sur le PC, lancez un programme d'émulation de terminal. Nous vous suggérons d'utiliser "OC-Console" présent sur le CD-ROM ou téléchargeable sur notre site Internet.
- Sélectionnez un débit de communication de 19200 Bds. Désactivez les protocoles de communication matériel et logiciel.
- Connectez une alimentation **stabilisée** (!) à la carte:
- GND sur le connecteur X7 broche 2
- +5V sur le connecteur X7 broche 1
- Contrôlez la tension, la polarité et l'absence de court-circuit **avant** de réaliser la connexion (sous peine de destruction de la carte, non pris en compte par la garantie)
- Une fois alimenté, le programme moniteur démarrera et affichera un message dans l'attente de vos commandes (voir p. 24 et 28).

Nous espérons que vous apprécierez de travailler avec la CardS12 !

3. Schéma de repérage des composants

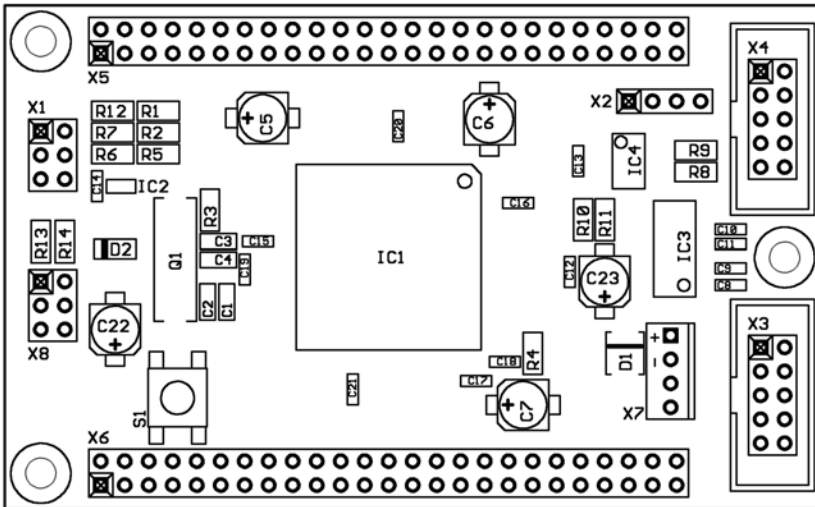
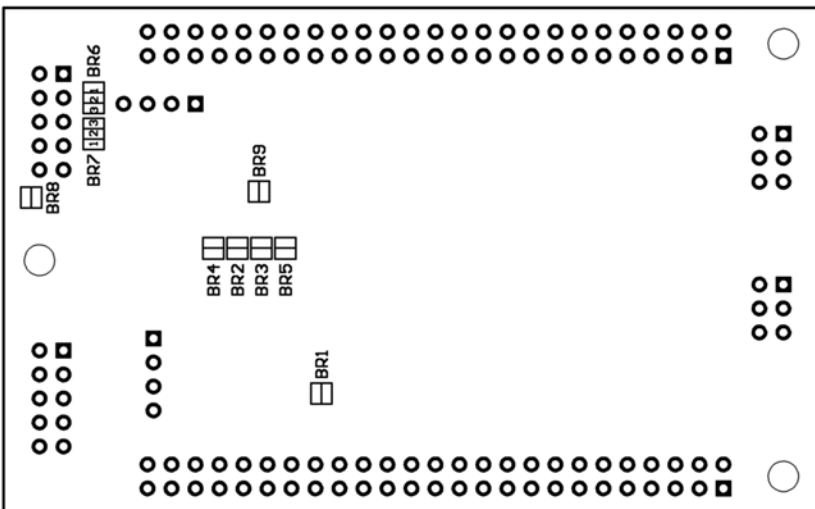


Schéma de la carte – Côté composants



Ponts de soudure de la carte (côté soudures)

4. Cavaliers et ponts de soudures

Cavaliers

Il n'y a aucun cavalier de configuration présent sur la carte.

Ponts de soudures

Sur le côté soudures de la carte, les ponts de soudure suivant peuvent être trouvés:

BR1: VRH

ouvert	alimentation externe de VRH
fermé *	VRH connecté à VDDA (VCC) sur la carte

BR2: T1IN

ouvert	Broche TXD0 (PS1) librement utilisable
fermé *	TXD0 connecté au transceiver RS232 IC3

BR3: T2IN

ouvert	Broche TXD1 (PS3) librement utilisable
fermé *	TXD1 connecté au transceiver RS232 IC3

BR4: R1OUT

ouvert	Broche RXD0 (PS0) librement utilisable
fermé *	RXD0 connecté au transceiver RS232 IC3

BR5: R2OUT

ouvert	Broche RXD1 (PS2) librement utilisable
fermé *	RXD1 connecté au transceiver RS232 IC3

* = Réglage d'usine par défaut

BR6, BR7: Sélection RS232 TxD/RxD (SER1)

- | | |
|------|--|
| 1-2* | RS232 configuré comme un "périphérique"
(connexion à un PC par exemple) |
| 2-3 | RS232 configuré comme "système hôte"
(connexion à un afficheur LCD série par exemple) |

BR8: Alimentation LCD (SER1)

- | | |
|----------|--|
| ouvert * | VCC non disponible sur port RS232 de SER1
(tracé connecteur Sub-D standard) |
| fermé | VCC disponible sur port RS232 de SER1
(sur la broche 9 du connecteur Sub-D) |

BR9: RXCAN0

- | | |
|--------|--|
| ouvert | Broche RXCAN0 (PM0) librement utilisable |
| fermé* | RXCAN0 connecté au transceiver CAN IC4 |

* = Réglage d'usine par défaut

5. Dimensions mécaniques

Le tableau ci-dessous résume les dimensions mécaniques de la CardS12. Les valeurs fournissent les bases de la conception de la carte.

Nota: Contrôlez toujours les dimensions mécaniques en utilisant une platine CardS12 réelle !

Le point de référence (0,0) est localisé au "sud/ouest" du coin de la carte. La carte est orientée à l'horizontal, comme représentée sur le schéma de la carte (voir ci avant dans la documentation).

Toutes les informations pour les trous de perçage (B) font référence au centre des trous de perçage. Le repérage des connecteurs (X) marque la broche 1.

	X en inch	Y en inch
X1	0,150	1,575
X2	2,600	1,700
X3	3,150	0,675
X4	3,150	1,825
X5	0,400	1,900
X6	0,400	0,100
X7	2,775	0,725
X8	0,150	0,950
B1	0,150	0,150
B2	0,150	1,950
B3	3,250	1,050
PCB	3,400	2,100

6. Description du circuit

Ce chapitre présente de nombreux détails sur la façon dont on doit procéder pour développer avec les HCS12 et avec la CardS12 en particulier.

Soyez s'il vous plait attentif au fait que même si ce manuel expose bon nombre d'indications spécifiques, il est impossible de couvrir toutes les techniques et connaissances nécessaires à la conception d'une application. A ce titre, consultez le data sheet des fabricants des microcontrôleurs et les manuels de votre logiciel de développement pour d'avantages d'informations.

Les programmes de démonstration présents dans ces chapitres ne sont donnés qu'à titre indicatif et nous ne pouvons garantir leurs exactitudes et pertinences dans le cadre d'une application spécifique.

Schéma théorique

Afin d'assurer une visibilité optimale, le schéma théorique de la CardS12 est livré sur un document séparé.

Cœur du contrôleur - Alimentation

La tension d'alimentation nominale de fonctionnement du MC9S12D64 est de 5 V. Ce MCU (IC1) a 3 paires de broches d'alimentation: VDDR/VSSR, VDDX/VSSX et VDDA/VSSA. Le cœur interne du MCU nécessite seulement 2.5V. La régulation nécessaire à la génération de cette tension est déjà intégrée dans le composant, aussi bien que celle des buffers d'"E/S" 5V dédiés aux broches d'entrées/sorties des ports. Vu de l'extérieur, le MCU sera donc considéré comme un composant 5 V. Il y a toutefois une exception: le signal de l'oscillateur et du PLL est issu de la tension du cœur du MCU et ne doit pas être piloté par des niveaux 5 V. Un niveau haut sur la broche VREGEN est nécessaire pour activer le régulateur interne.

Les trois paires de connexions mentionnées ci-avant doivent être découplées avec la plus grande attention. Un condensateur céramique d'au moins 100nF doit être connecté directement sur chaque paire (C15, C16, C17). Il est recommandé d'ajouter un condensateur

électrochimique (ou tantale goutte) de 10 μ F à chaque connexion et ce plus particulièrement si certaines broches des ports du MCU doivent être fortement chargées (C5, C6, C7). Une attention toute particulière doit être apportée avec VDDA, puisqu'il s'agit du point de référence (VDDA/2) pour le régulateur de tension interne.

La tension de référence du cœur apparaît aux paires de broches VDD1/VSS1, VDD2/VSS2 et VDDPLL/VSSPLL, qui doivent également être découplées (C19, C20, C21). Ces broches ne doivent pas être utilisées pour alimenter un dispositif externe. Ceci est plus particulièrement vrai pour VDDPLL, qui sert de point de référence pour la boucle du filtre du PLL (R3, C3, C4).

Il y a deux broches du MCU (VRH/VRL) qui permettent de déterminer les limites inférieures et supérieures du convertisseur analogique/numérique (ATD). Tandis que VRL est connecté à la masse, VRH est connecté à VDDA via un pont de soudure BR1. C18 est utilisé pour le découplage. VHR peut être alimenté de l'extérieur après avoir ouvert le pont de soudure BR1. Ceci peut être utile si la tension d'alimentation principale ne répond pas aux tolérances nécessaires ou si ATD doit fonctionner avec une valeur de référence inférieure à 5 V. VRH ne doit pas excéder VDDA, eu regard au mode d'alimentation sélectionné.

La broche TEST est utilisée uniquement lors des tests de fabrication. Cette broche doit être tout le temps connectée à la masse.

Génération du Reset

/RESET est la broche bi-directionnelle active à l'état bas du MCU's. En tant qu'entrée, elle initialise le MCU de façon asynchrone vis-à-vis d'un état de démarrage connu. En tant que sortie "drain ouvert", elle indique qu'une condition de reset (MCU interne) a été déclenchée. Le MCU du HCS12 dispose déjà d'un circuit interne de génération de reset gérant le reset à la mise sous tension, d'un timer watchdog (COP) et d'un moniteur d'horloge. Il est toutefois nécessaire d'ajouter un circuit d'inhibition basse tension (LCI), aussi appelé "contrôleur de reset". La tâche de ce circuit est de générer une condition de reset stable si la tension d'alimentation descend en dessous du niveau nécessaire à un fonctionnement correct du MCU.

Afin de prévenir toute "collision" avec la broche bi-directionnelle /RESET du MCU, le circuit LVI IC2 dispose d'une sortie sur "drain ouvert". A l'état inactif, elle est ramenée à l'état haut via la résistance R6. Le seuil de détection du IC2 est typiquement de 4,6 V, ce qui est légèrement supérieur au niveau de tension minimum requis pour le MCU (4,5 V).

Par ailleurs IC2 est capable de prolonger la sortie reset pour filtrer efficacement les impulsions courtes sur la tension d'alimentation. La durée de ce délai peut être définie par la capacité C14. Une valeur de 100 nF permet d'obtenir un délai d'environ 50 .. 80 ms.

Il est important de souligner que ce délai ne sera appliqué que lors du cycle d'alimentation. IC2 ne prolongera pas les impulsions suite à un RESET généré en interne par le MCU. Ceci est important, sans quoi le MCU ne serait plus en mesure de détecter la source du reset. Dès lors ceci pourrait conduire à un mauvais vecteur de reset pouvant causer un crash logiciel. Sachez également qu'un condensateur sur la ligne du reset pourrait causer les mêmes conditions fatales et c'est pourquoi les circuits externes connectés à la broche /RESET des HC12/HCS12 ne doivent jamais être dotés d'une capacité de grande valeur !

Génération horloge et PLL

Le circuit d'oscillation intégré au MC9S12D64 peut générer l'horloge primaire (OSCCLK) en utilisant un quartz (Q1) connecté entre les broches EXTAL et XTAL. La fréquence autorisée est comprise entre 0.5 à 16MHz. Comme de bien évident, 2 condensateurs doivent être associés au quartz (C1, C2). Toutefois ce montage est différent de celui d'un oscillateur de "Pierce" standard qui est communément utilisé sur les HC11 et HC12 (et a été ajouté aux nouveaux HCS12 comme une option de configuration de l'oscillateur).

Sur la CardS12, le MC9S12D64 utilise un oscillateur de "Colpitts" modifié. Le principale avantage est une consommation réduite, bien que la sélection des composants soit plus critique. Le circuit de la CardS12 utilise un quartz 16 MHz de grande qualité combiné à des capacités de seulement 3,9 pF. Par ailleurs, un soin tout particulier a été pris lors de la conception afin d'éviter autant que faire ce peu l'introduction d'une capacité parasite entre XTAL et EXTAL.

Avec une horloge OSCCLK de 16MHz, la vitesse de bus interne (ECLK) est de 8 MHz par défaut. Pour obtenir une vitesse supérieure, le PLL doit être activé. Le MC9S12D64 peut fonctionner avec une vitesse de bus pouvant aller jusqu'à 25MHz, toutefois la plupart des applications se limitent à 24 MHz car cette valeur est plus appropriée pour générer une large gamme de débits SCI (bauds) standardisés.

Une boucle de filtrage passif doit être placée sur la broche XFC. Le filtre passe-bas (R3, C3, C4) est de second ordre pour éliminer l'oscillation d'entrée du VCO. La valeur du réseau du filtre extérieur et la référence de fréquence déterminent la vitesse de correction et la stabilité du PLL. Si le PLL n'est pas utilisé, la broche XFC doit être appliquée à VDDPLL.

Le choix des valeurs des composants du filtre doit toujours être un compromis entre le temps de verrouillage et la stabilité de la boucle. Une bande passante de 5 à 10 KHz et un facteur d'amortissement de 0,9 sont un bon point de départ pour les calculs. Avec un quartz de 16 MHz et une horloge bus de 24 MHz, on obtient les valeurs: $R3 = 4.7k$ et $C3 = 22nF$. $C4$ devra être approximativement $(1/20..1/10) \times C3$, ex.: $2.2nF$ dans notre cas. Référez-vous s'il vous plaît au "XFC Component Selection" dans le guide d'utilisation du MC9S12DP256B pour avoir plus d'informations sur la façon de calculer la valeur du filtre pour d'autres configurations. Le fichier source ci-après montre les étapes nécessaires pour initialiser le PLL:

```
//=====
// File: s12_crg.c - v1.00
//=====

/-- Includes -----
#include <hcs12dp256.h>
#include "s12_crg.h"

/-- Code -----

void initPLL(void) {

    CLKSEL &= ~BM_PLLSEL;           // make sure PLL is *not* in use
    PLLCTL |= BM_PLLON+BM_AUTO;     // enable PLL module, Auto Mode
    REFDV = S12_REFDV;              // set up Reference Divider
    SYNCR = S12_SYNR;               // set up Synthesizer Multiplier
    // the following dummy write has no effect except consuming some cycles,
    // this is a workaround for erratum MUCTS00174 (mask set 0K36N only)
    // CRGFLG = 0;
    while((CRGFLG & BM_LOCK) == 0) ; // wait until PLL is locked
    CLKSEL |= BM_PLLSEL;            // switch over to PLL clock
}

//=====
```

Il est également possible d'utiliser une source d'horloge externe pour le MC9S12D64 si l'oscillateur interne et le PLL sont désactivés en appliquant un niveau bas sur la /XCLKS pendant le reset. Du fait que cette option ne soit pas utilisée par défaut sur la CardS12, la broche /XCLKS est maintenue au niveau haut via une résistance de tirage interne au MCU. Notez toutefois que d'autres modèles de HCS12 peuvent avoir une gestion différente des fonctionnalités liées à leur broche /XCLKS.

Modes opératoires, Support BDM

Trois broches du HCS12 sont utilisées pour sélectionner le mode opératoire du MCU: MODA, MODB et BKGD (=MODC). Tandis que MODA et MODB sont ramenées à la masse (R1, R2) pour sélectionner le mode "Single Chip", BKGD est forcée au niveau haut (R7) par défaut. Dans ces conditions, le MCU démarrera normalement en mode "Single Chip", qui correspond au mode le plus communément utilisé pour les applications utilisant un HCS12.

Le mode opératoire du HCS12 utilisé pour le téléchargement et le débogage est appelé Background Debug Mode (BDM). BDM est actif immédiatement après un reset si les broches de sélection de mode MODA/MODB/BKGD sont configurées pour le mode Special Single Chip. Ceci peut être fait en forçant la broche BKGD au niveau bas pendant le reset, tandis que les broches MODA et MODB sont également au niveau bas.

Parce que seul le niveau logique de BKGD diffère entre ces 2 modes, il est très simple d'opérer cette modification. Toutefois, il n'y a pas besoin de prévoir un cavalier ou un pont de soudure car cette tâche peut être faite par un boîtier BDM-Pod (tel que le ComPOD12) relié au connecteur X1. Le BDM-Pod étant de toute façon nécessaire pour les opérations de téléchargement et / ou débogage via BDM, il est donc aisé pour ce dernier de prendre automatiquement cette fonction en charge sous la commande du logiciel associé sur PC.

Convertisseur A/N intégré

Le MC9S12D64 dispose de 2 modules de conversion Analogique / Numérique 10 bits. Chaque module (ATD0, ATD1) dispose de huit canaux d'entrées multiplexés. VRH est la tension de référence de niveau haut pour tous les canaux A/N. Sur la CardS12, VRH est connecté à VDDA (5 V) au travers d'un pont de soudure BR1. En ouvrant BR1, il est possible d'utiliser une tension de référence externe.

L'exemple de programme ci-dessous montre la séquence d'initialisation du module de conversion A/N (ATD0) ainsi qu'une routine de conversion d'un seul canal. Le fichier source S12_ATD.C contient également des fonctions additionnelles pour le module ATD.

```
//=====
// File: S12_ATD.C - V1.00
//=====

//-- Includes -----
#include "datatypes.h"
#include <hcs12dp256.h>
#include "s12_atd.h"

//-- Code -----

// Func: Initialize ATD module
// Args: -
// Retn: -
//
void initATD0(void) {

    // enable ATD module
    ATD0CTL2 = BM_ADP;
    // 10 bit resolution, clock divider=12 (allows ECLK=6..24MHz)
    // 2nd sample time = 2 ATD clocks
    ATD0CTL4 = BM_PRS2 | BM_PRS0;
}

//-----

// Func: Perform single channel ATD conversion
// Args: channel = 0..7
// Retn: unsigned, left justified 10 bit result
//
UINT16 getATD0(UINT8 channel) {

    // select one conversion per sequence
    ATD0CTL3 = BM_S1C;
    // right justified unsigned data mode
    // perform single sequence, one out of 8 channels
    ATD0CTL5 = BM_DJM | (channel & 0x07);
    // wait until Sequence Complete Flag set
    // CAUTION: no loop time limit implemented!
    while((ATD0STAT0 & BM_SCF) == 0) ;
    // read result register
    return ATD0DR0;
}

//-----
```


Mémoire EEPROM intégrée

Le module mémoire EEPROM interne au MC9S12D64 renferme 1 KB. Il est composé de 256 secteurs de 4 octets (32 bits) par secteur. Pour l'effacement, n'importe quel secteur peut être sélectionné. La programmation est effectuée en mode "word" (2 octets). La lecture peut être effectuée en mode "word" ou octet.

Après un reset, le module EEPROM du MC9S12D64 est consigné à *la fois* à l'adresse 0x0000 *et* (en même temps) à l'adresse 0x0400. Toutefois, dans l'espace bas (0x0000..0x03FF), les registres de contrôle sont prioritaires sur l'EEPROM. Le module EEPROM peut être réalloué à n'importe quelle limite des 2 KB (Voir registre de contrôle INITEE).

Dans l'exemple qui suit, le module EEPROM est laissé dans la position par défaut. La séquence d'initialisation prend seulement en charge les réglages du diviseur d'horloge EEPROM en fonction de la fréquence du quartz. La fonction d'écriture wrSectEETS() copie deux mots (4 octets) depuis l'adresse source src à l'adresse EEPROM dest. dest doit être identique à l'emplacement en EEPROM (alignement d'une valeur de 32 bits). Si le secteur n'est pas effacé (état effacé = 0xFFFFFFFF), la routine réalisera un effacement du secteur avant d'écrire dans ce dernier.

Les fonctions d'accès readItemEETS() et writelItemEETS() permettent d'obtenir une fonction plus abstraite sur la façon dont on peut gérer le contenu de l'EEPROM. Plutôt que d'utiliser certaines adresses qui doivent faire partie de la plage d'adresse de l'EEPROM, ces routines utilisent des "nombres abstraits", qui pour chacun d'eux consiste en une quantité variable de données (1 à 4 octets).

```

//=====
// File: S12_EETS.C - V1.00
//=====

/-- Includes -----

#include "datatypes.h"
#include <hcs12dp256.h>
#include "s12_eets.h"

/-- Code -----

void initEETS(void) {

    ECLKDIV = EETS_ECLKDIV;          // set EEPROM Clock Divider Register
}

//-----

INT8 wrSectEETS(UINT16 *dest, UINT16 *src) {

    // check addr: must be aligned 32 bit
    if((UINT16)dest & 0x0003) return -1;
    // check if ECLKDIV was written
    if((ECLKDIV & BM_EDIVLD) == 0) return -2;
    // make sure error flags are reset
    ESTAT = BM_PVIOL | BM_ACCERR;
    // check if command buffer is ready
    if((ESTAT & BM_CBEIF) == 0) return -3;
    // check if sector is erased
    if((*dest != 0xffff) || (*(dest+1) != 0xffff)) {
        // no, go erase sector
        *dest = *src;
        ECMD = EETS_CMD_SERASE;
        ESTAT = BM_CBEIF;
        if(ESTAT & (BM_PVIOL | BM_ACCERR)) return -4;
        while((ESTAT & BM_CBEIF) == 0) ;
    }

    // program 1st word
    *dest = *src;
    ECMD = EETS_CMD_PROGRAM;
    ESTAT = BM_CBEIF;
    if(ESTAT & (BM_PVIOL | BM_ACCERR)) return -5;
    while((ESTAT & BM_CBEIF) == 0) ;
    // program 2nd word
    *(dest+1) = *(src+1);
    ECMD = EETS_CMD_PROGRAM;
    ESTAT = BM_CBEIF;
    if(ESTAT & (BM_PVIOL | BM_ACCERR)) return -6;
    while((ESTAT & BM_CBEIF) == 0) ;
    return 0;
}

//-----

INT8 writeItemEETS(UINT16 item_no, void *item) {

    if(item_no >= EETS_MAX_SECTOR) return -7;
    item_no = EETS_START + (item_no << 2);
    return wrSectEETS((UINT16 *)item_no, (UINT16 *)item);
}

//-----

INT8 readItemEETS(UINT16 item_no, void *item) {

    if(item_no >= EETS_MAX_SECTOR) return -7;
    item_no = EETS_START + (item_no << 2);
    *((UINT16 *)item) = *((UINT16 *)item_no);
    *((UINT16 *)item)+1 = *((UINT16 *)item_no)+1;
    return 0;
}

//=====

```

Témoin LED

La broche PH7 pilote un témoin LED (D2). Pour contrôler cette LED, des macros simples peuvent être utilisés comme montré ci-dessous:

```
//=====
// File: CARDS12_LED.H - V1.00
//=====

#ifndef __CARDS12_LED_H
#define __CARDS12_LED_H

//-- Macros -----

#define initLED()    PORTH |= 0x80; DDRH |= 0x80
#define offLED()     PORTH |= 0x80
#define onLED()      PORTH &= ~0x80
#define toggleLED()  PORTH ^= 0x80

//-- Function Prototypes -----

/* module contains no code */

#endif // __CARDS12_LED_H =====
```

Interface RS-232

Le MC9S12D64 dispose de deux interfaces séries asynchrones (SCI0, SCI1). Chaque interface possède une ligne de réception et une ligne d'émission (RXDx, TXDx). Les signaux de contrôle (Handshake) ne sont pas fournis par le module; ils peuvent être ajoutés en utilisant des ports d'entrées/sorties si nécessaires.

Sur la CardS12, le signal des deux SCI est relié à un circuit transceiver industriel (IC3) au standard RS-232 au travers de ponts de soudure (BR2..BR5), qui sont établis par défaut. En ouvrant ces derniers, les signaux du contrôleur peuvent être utilisés pour d'autres usages. Il est possible d'accéder aux signaux via le connecteur X6.

X3 (SCI0) est utilisé comme interface primaire RS-232. Pour connecter la CardS12 à un PC, un câble plat à 10 conducteurs peut être utilisé. Le câble doit être équipé d'un connecteur femelle 10 broches côté CardS12 (X3) et femelle Sub-D 9 broches côté du PC.

Ceci est également valable pour X4 (SCI1), à condition que BR6 et BR7 soient en position 1-2. Dans ce cas, le PC servira d'hôte et la CardS12 sera configurée en périphérique.

La configuration inverse peut être utilisée pour connecter un afficheur LCD à commande série sur X4. Dans ce cas, la CardS12 sera l'hôte et le LCD un périphérique. Le signal de croisement nécessaire est effectué en changeant la position de BR6 et BR7 en 2-3. En plus, il peut être utile de fermer BR8 afin d'alimenter le module LCD via la broche 9 du connecteur Sub-D 9 broches (Attention: ceci n'est pas conforme avec le standard RS-232 !).

Les afficheurs à commandes séries sont proposés par de nombreux fabricants tel que la société Canadienne Matrix Orbital (<http://www.matrixorbital.com>).

L'exemple de code ci-dessous montre comment utiliser SCI0 en mode polling.

```
//=====
// File: S12_SCI.C - V1.00
//=====

/-- Includes -----
#include "datatypes.h"
#include <hcs12dp256.h>
#include "s12_sci.h"

/-- Code -----

void initSCI0(UINT16 bauddiv) {
    SCI0BD = bauddiv & 0x1fff; // baudrate divider has 13 bits
    SCI0CR1 = 0; // mode = 8N1
    SCI0CR2 = BM_TE+BM_RE; // Transmitter + Receiver enable
}

//-----

UINT8 getSCI0(void) {
    while((SCI0SR1 & BM_RDRF) == 0) ;
    return SCI0DRL;
}

//-----

void putSCI0(UINT8 c) {
    while((SCI0SR1 & BM_TDRE) == 0) ;
    SCI0DRL = c;
}

//-----
```

Bus SPI

Le MC9S12D64 dispose d'un module SPI (SPI0), qui peut être utilisé pour des communications séries synchrones avec des composants SPI externes.

SPI0 est composé de quatre signaux individuels: MISO, MOSI, SCK et /SS (broches PS4 à PS7 du MCU). Ces signaux se sont pas utilisés sur la CardS12, mais sont accessibles au travers des connecteurs en bout de carte.

Le listing suivant montre quelques fonctions de base (initialisation, transfert de données 8 bits) pour le port SPI0 (la prise en compte du signal chip select n'est pas pris en compte):

```
//=====
// File: s12_SPI.C - V1.00
//=====

//-- Includes -----

#include "datatypes.h"
#include <hcs12dp256.h>
#include "s12_spi.h"

//-- Code -----

void initSPI0(UINT8 bauddiv, UINT8 cpol, UINT8 cpha) {

    DDRS |= 0xe0;                // SS,SCK,MOSI Output
    SPI0BR = bauddiv;            // set SPI Rate
    // enable SPI, Master Mode, select clock polarity/phase
    SPI0CR1 = BM_SPE | BM_MSTR | (cpol ? BM_CPOL : 0) | (cpha ? BM_CPHA : 0);
    SPI0CR2 = 0;                // as default
}

//-----

UINT8 xferSPI0(UINT8 abyte) {

    SPI0DR = abyte;              // start transfer
    while((SPI0SR & BM_SPIF) == 0) ; // wait until transfer finished
    return(SPI0DR);              // read back data received
}

//=====
```

Bus IIC

Les broches PJ6 et PJ7 permettent l'accès au module Inter-IC-Bus (IIC/I2C/I²C) du MC9S12D64. Du fait que le Bus IIC soit implémenté en hardware, une émulation logiciel IIC devient superflue.

Pour les 2 signaux du bus IIC (SDA, SCL), les résistances de tirage au niveau haut sont nécessaires. Elles peuvent être équipées sur la carte (R10, R11) ou fournies extérieurement.

Les signaux du Bus IIC sont disponibles sur X5/47+48.

Interface CAN

Le MC9S12D64 dispose d'un module CAN, référencé CAN0.

CAN0 utilise les broches PM0 et PM1. IC4 sert de bus d'interface CAN physique. C'est un circuit d'interface haute vitesse très communément utilisé dans les applications industrielles. R9 détermine les réglages de contrôle de rampe (à modifier pour communication haute vitesse, voir data-sheet). R8 est une résistance de terminaison, qui devient nécessaire si la CardS12 est le dernier maillon de la chaîne CAN. Fermez la connexion entre les broches 1 et 2 de X2 dans ce cas, sinon laissez le ouvert.

Si CAN0 n'est pas utilisé, BR9 peut être ouvert afin que la broche PM0 du MCU puisse être utilisée en "entrée/sortie" pour d'autres applications. Il peut être accessible sur X5/41 (et PM1 à X5/42).

7. Conseils pour vos applications

Comportement après Reset

Dès que l'entrée de reset du microcontrôleur est libérée, le MCU lit le vecteur d'interruption à l'adresse mémoire \$FFFE/F et ensuite va à l'adresse qu'il a trouvé.

Lorsqu'elle vient de vous être livrée, la mémoire Flash de la CardS12 renferme un programme moniteur TwinPEEKs. Le vecteur reset pointe vers l'adresse de début de ce moniteur. Il en résulte que le moniteur démarrera immédiatement après une condition de reset.

Code de démarrage

Tout programme de microcontrôleur démarre par une série de commandes d'initialisations matériel. Pour la CardS12, seul la configuration du pointeur de pile est primordiale. Tandis qu'il est important sur la plupart des produits HC12 de désactiver le Watchdog, le contrôleur de Watchdog des produits HCS12 est déjà désactivé en dehors du reset.

Informations additionnelles sur le Web

Des informations additionnelles au sujet de la carte CardS12 seront publiées sur notre site Web. Lorsqu'elle seront dispos connectez-vous à l'adresse:

<http://elmicro.com/cards12.html>

8. Moniteur TwinPEEKs

Version logiciel 2.0

Communication Série

Le moniteur TwinPEEKs communique avec la première interface série RS-232 ("SER0", X3) à **19200 Baud**. Les réglages sont: 8N1, sans contrôle de flux matériel (ni logiciel) et sans protocole.

Fonction Autostart

Après un reset, le moniteur TwinPEEKs vérifie si les ports PH6 et PH7 sont connectés. Si c'est le cas, le moniteur démarre automatiquement à l'adresse \$8000.

Cette caractéristique permet de démarrer un programme d'application automatiquement sans avoir à modifier le vecteur d'interruption qui est logé dans le block protégé de boot flash.

Accès en écriture à la Flash et l'EEPROM

Le CPU peut lire chaque octet des ressources du microcontrôleur – Le type de mémoire n'a pas d'importance. Toutefois, dans le cas d'une écriture, quelques règles doivent être suivies: La Flash et l'EEPROM doivent être effacées avant chaque procédure d'écriture. La programmation doit être faite en mode "word" (deux octets à la fois) pour aligner les adresses.

Pour mettre en forme ces données, deux octets consécutifs doivent être combinés (le moniteur TwinPEEKs est en attente de ce format). Toutefois le problème qui suit ne peut pas être évité par le moniteur:

Le moniteur traite chaque ligne du fichier S-Record séparément. Si la dernière adresse du fichier S-Record est paire, le second octet servant à réaliser le "mot" est manquant. Le moniteur TwinPEEKs ajoutera alors l'octet \$FF dans ce cas, afin qu'il soit capable d'effectuer l'écriture du "mot".

Le problème apparaît, si l'octet de la chaîne est présent à la ligne suivante du fichier S-Record. L'octet qui était manquant lors de

l'opération précédente nécessitera une seconde écriture à la même adresse (mot) – ce qui ne sera pas permis. Avec comme principale conséquence, l'apparition d'un message d'erreur ("non effacé").

Pour éviter ce problème, il est nécessaire d'aligner toutes les données du fichier S-Record avant la programmation. Ceci peut être fait en utilisant l'outil gratuit de Motorola™ SRECCVT:

```
SRECCVT -m 0x00000 0xffff 32 -o <outfile> <infile>
```

Une description détaillée de ce logiciel est disponible dans le Guide de référence SRECCVT (format PDF).

Note: Il n'est pas possible de programmer ou d'effacer la partie de la mémoire Flash qui contient le code du moniteur. De même, les derniers 16 octets du block de l'EEPROM sont réservés pour le fonctionnement du système.

Ré-acheminement des Vecteurs d'interruption

Les vecteurs d'interruption des HCS12 sont logés en fin de la plage mémoire des 64 KB, dans laquelle se trouve le code protégé du moniteur. Il en résulte que votre application ne pourra pas modifier ces vecteurs d'interruption. Afin de palier à ce problème, le moniteur ré-acheminera tous ces vecteurs en RAM (à l'exception du vecteur de Reset). La procédure est similaire à celle utilisée sur le mode spécial Bootstrap des HC11.

Le programme d'application peut modifier les vecteurs d'interruption durant son exécution (avant l'autorisation d'interruption générale) en plaçant des instructions de "saut" en RAM à l'emplacement considéré). L'exemple qui suit montre les étapes à suivre pour utiliser l'interruption IRQ.

```
ldaa #$06          ; JMP opcode to
staa $3FEE         ; IRQ pseudo vector
ldd  #isrFunc      ; ISR address to
std  $3FEF         ; IRQ pseudo vector + 1
```

Pour un programme en C, la séquence qui suit peut être utilisée:

```
// install IRQ pseudo vector in RAM
// (if running with TwinPEEKs monitor)
*((unsigned char *)0x3fee) = 0x06;    // JMP opcode
*((void (**)(void))0x3fef) = isrFunc;
```

Le listing suivant est une partie du programme moniteur. Il montre les adresses de vecteurs originaux (1^{er} colonne depuis la gauche) avec les adresses ré-acheminées en RAM (2^{ème} colonne):

FF80 : 3F43	dc.w	TP_RAMTOP-189	; reserved
FF82 : 3F46	dc.w	TP_RAMTOP-186	; reserved
FF84 : 3F49	dc.w	TP_RAMTOP-183	; reserved
FF86 : 3F4C	dc.w	TP_RAMTOP-180	; reserved
FF88 : 3F4F	dc.w	TP_RAMTOP-177	; reserved
FF8A : 3F52	dc.w	TP_RAMTOP-174	; reserved
FF8C : 3F55	dc.w	TP_RAMTOP-171	; PWM Emergency Shutdown
FF8E : 3F58	dc.w	TP_RAMTOP-168	; Port P
FF90 : 3F5B	dc.w	TP_RAMTOP-165	; CAN4 transmit
FF92 : 3F5E	dc.w	TP_RAMTOP-162	; CAN4 receive
FF94 : 3F61	dc.w	TP_RAMTOP-159	; CAN4 errors
FF96 : 3F64	dc.w	TP_RAMTOP-156	; CAN4 wake-up
FF98 : 3F67	dc.w	TP_RAMTOP-153	; CAN3 transmit
FF9A : 3F6A	dc.w	TP_RAMTOP-150	; CAN3 receive
FF9C : 3F6D	dc.w	TP_RAMTOP-147	; CAN3 errors
FF9E : 3F70	dc.w	TP_RAMTOP-144	; CAN3 wake-up
FFA0 : 3F73	dc.w	TP_RAMTOP-141	; CAN2 transmit
FFA2 : 3F76	dc.w	TP_RAMTOP-138	; CAN2 receive
FFA4 : 3F79	dc.w	TP_RAMTOP-135	; CAN2 errors
FFA6 : 3F7C	dc.w	TP_RAMTOP-132	; CAN2 wake-up
FFA8 : 3F7F	dc.w	TP_RAMTOP-129	; CAN1 transmit
FFAA : 3F82	dc.w	TP_RAMTOP-126	; CAN1 receive
FFAC : 3F85	dc.w	TP_RAMTOP-123	; CAN1 errors
FFAE : 3F88	dc.w	TP_RAMTOP-120	; CAN1 wake-up
FFB0 : 3F8B	dc.w	TP_RAMTOP-117	; CAN0 transmit
FFB2 : 3F8E	dc.w	TP_RAMTOP-114	; CAN0 receive
FFB4 : 3F91	dc.w	TP_RAMTOP-111	; CAN0 errors
FFB6 : 3F94	dc.w	TP_RAMTOP-108	; CAN0 wake-up
FFB8 : 3F97	dc.w	TP_RAMTOP-105	; FLASH
FFBA : 3F9A	dc.w	TP_RAMTOP-102	; EEPROM
FFBC : 3F9D	dc.w	TP_RAMTOP-99	; SPI2
FFBE : 3FA0	dc.w	TP_RAMTOP-96	; SPI1
FFC0 : 3FA3	dc.w	TP_RAMTOP-93	; IIC
FFC2 : 3FA6	dc.w	TP_RAMTOP-90	; BDLC
FFC4 : 3FA9	dc.w	TP_RAMTOP-87	; Self Clock Mode
FFC6 : 3FAC	dc.w	TP_RAMTOP-84	; PLL Lock
FFC8 : 3FAF	dc.w	TP_RAMTOP-81	; Pulse Accu B Overflow
FFCA : 3FB2	dc.w	TP_RAMTOP-78	; MDCU
FFCC : 3FB5	dc.w	TP_RAMTOP-75	; Port H
FFCE : 3FB8	dc.w	TP_RAMTOP-72	; Port J
FFD0 : 3FBB	dc.w	TP_RAMTOP-69	; ATD1
FFD2 : 3FBE	dc.w	TP_RAMTOP-66	; ATD0
FFD4 : 3FC1	dc.w	TP_RAMTOP-63	; SC11
FFD6 : 3FC4	dc.w	TP_RAMTOP-60	; SC10
FFD8 : 3FC7	dc.w	TP_RAMTOP-57	; SPI0
FFDA : 3FCA	dc.w	TP_RAMTOP-54	; Pulse Accu A Input Edge
FFDC : 3FCD	dc.w	TP_RAMTOP-51	; Pulse Accu A Overflow
FFDE : 3FD0	dc.w	TP_RAMTOP-48	; Timer Overflow

FFE0 : 3FD3	dc.w	TP_RAMTOP-45	; TC7
FFE2 : 3FD6	dc.w	TP_RAMTOP-42	; TC6
FFE4 : 3FD9	dc.w	TP_RAMTOP-39	; TC5
FFE6 : 3FDC	dc.w	TP_RAMTOP-36	; TC4
FFE8 : 3FDF	dc.w	TP_RAMTOP-33	; TC3
FFEA : 3FE2	dc.w	TP_RAMTOP-30	; TC2
FFEC : 3FE5	dc.w	TP_RAMTOP-27	; TC1
FFEE : 3FE8	dc.w	TP_RAMTOP-24	; TC0
FFF0 : 3FEB	dc.w	TP_RAMTOP-21	; RTI
FFF2 : 3FEE	dc.w	TP_RAMTOP-18	; IRQ
FFF4 : 3FF1	dc.w	TP_RAMTOP-15	; XIRQ
FFF6 : 3FF4	dc.w	TP_RAMTOP-12	; SWI
FFF8 : 3FF7	dc.w	TP_RAMTOP-9	; Illegal Opcode
FFFA : 3FFA	dc.w	TP_RAMTOP-6	; COP Fail
FFFC : 3FFD	dc.w	TP_RAMTOP-3	; Clock Monitor Fail
FFFE : F000	dc.w	main	; Reset

Utilisation

Une commande TwinPEEKs se compose d'un caractère, suivi par un ou plusieurs arguments (si nécessaire). Tous les nombres sont hexadécimaux sans préfixe, ni suffixe. Les minuscules et les majuscules sont reconnues.

La plage mémoire visible du CPU est de 64 KB, il en résulte que les arguments des adresses ne sont pas plus longs que 4 digits. La fin d'une adresse est toujours exclue des commandes. Par exemple la commande "D 1000 1200" affichera la plage mémoire depuis \$1000 jusqu'à \$11FF.

Les données de saisie sont contenues dans le buffer de la ligne du moniteur. Les codes ASCII valides sont compris entre \$20 à \$7E. La touche de retour (Backspace - \$08) effacera les caractères à gauche du curseur. La touche <ENTER> (\$0A) est utilisée pour valider la commande saisie.

Le "prompt" du moniteur affiche toujours la page mémoire en cours d'utilisation (c'est à dire le contenu du registre PPAGE).

Commandes du Moniteur

Test virginité (Blank Check)

Syntaxe: B

Teste la virginité de toute la mémoire Flash (sauf code mémoire du moniteur). Si la mémoire Flash n'est pas vierge, alors le numéro de la première page ne contenant pas un octet à \$ FF sera affiché.

Affichage contenu mémoire (Dump Memory)

Syntaxe: D [adr1 [adr2]]

Affiche le contenu de la mémoire depuis l'adresse adr1 jusqu'à l'adresse adr2. Si le paramètre adr2 n'est pas indiqué, l'affichage montrera les \$40 octets suivant l'adresse de début adr1. L'octet présent à l'adresse adr1 sera affiché entre les caractères <>.

Modification mémoire (Edit Memory)

Syntax: **E** [**addr** {**byte**}]

Dans la ligne de commande, l'adresse de début peut être suivie d'un à quatre octets, permettant une modification d'octets, de données type "word" et "double word". La modification est immédiate et l'écran du PC affichera le "prompt" après l'opération.

Si la ligne de commande ne contient aucun octet, le moniteur entre dans le mode de modification interactif. Le moniteur sera capable d'identifier les emplacements mémoire (Flash/EEPROM) pour lesquels seule la modification de données type "word" est possible. Dans de tels cas, le moniteur attendra la saisie d'une donnée sur 16 bits.

Pour sortir du mode de modification interactif, tapez simplement la touche "Q". Les commandes du mode de modification interactif sont:

```
<ENTER>  prochaine adresse
-         adresse précédente
=         même adresse
.         Sortie du mode (idem Q)
```

Remplissage mémoire (Fill Memory)

Syntaxe: **F** **adr1** **adr2** **byte**

Remplissage de la mémoire depuis l'adresse adr1 jusqu'avant la fin de adr2 avec la valeur "byte".

Saut à une adresse (Goto Address)

Syntaxe: **G** [**addr**]

Appel le programme d'application présent à l'adresse addr. Nota: il n'y a pas de moyen de revenir au programme moniteur après coup.

Aide (Help)

Syntaxe: **H**

Affiche la liste des commandes reconnues par le moniteur.

Info Système (System Info)

Syntaxe: I

Affiche les informations sur l'organisation des plages mémoires des registres de bloc, de RAM, d'EEPROM et de Flash ainsi que l'identifiant du MCU (PARTID).

Charger (Load)

Syntaxe: L

Charger un fichier S-Record en mémoire. Les enregistrements de données de type S1 (16-bit MCU addresses) et S2 (linear 24-bit addresses) peuvent être utilisés.

Les enregistrements S0 (lignes de commentaires) seront sautés.

Les enregistrement de type S8- et S9 sont reconnus en temps que fin de fichier.

Les enregistrements de type S2- utilisent des adresses linéaires comme stipulé par les indications de Motorola™. Les plages d'adresses autorisées pour le MC9S12D64 vont de 0xF0000 (0x3C * 16KB) jusqu'à 0xFFFFF (0x40 * 16 KB - 1).

Avant de charger les données en mémoires non volatiles (EEPROM, Flash), il vous faudra au préalable toujours effacer celles-ci. Il vous faudra également vous assurer que les données soient au bon format avant le chargement (voir explication des chapitres précédents).

Après avoir tapé la touche L, le moniteur affiche le message "Loading...", cliquez alors sur le bouton "Transfert", sélectionnez le fichier (au bon format) et cliquez sur le bouton "Transmit". Le terminal de saisie (tel OC-Console) attendra un accusé de réception (caractère *) de la part de la carte avant de lui envoyer une nouvelle ligne. Cette méthode permet une synchronisation entre la vitesse du PC et celle de la carte (MCU).

Déplacer Mémoire (Move Memory)**Syntaxe:** **M** adr1 adr2 adr3

Copier un bloc mémoire depuis l'adresse adr1 jusqu'à adr2 (non inclus) à l'emplacement démarrant à l'adresse adr3.

Sélection PPAGE (Select PPAGE)**Syntaxe:** **P** [page]

Sélectionne une page programme (PPAGE). Cette page deviendra visible dans une fenêtre de 16 KB de &8000 à \$BFFF.

Effacer Flash (Erase Flash)**Syntaxe:** **X** [page]

Efface une page (16KB) de mémoire Flash.

Si le paramètre page n'est pas indiqué, toute la mémoire Flash sera effacée (sauf le code du moniteur) après demande de confirmation. Pour effacer le code du moniteur, il vous faudra passer par un outil de programmation via le connecteur BDM (comme par exemple le ComPOD12/StarProg).

Effacer EEPROM (Erase EEPROM)**Syntaxe:** **Y** [sadr]

Efface un secteur (double "word" = 4 octets) de mémoire EEPROM. Le secteur est spécifié par son adresse de début sadr (les bits 0 et 1 de sadr ne sont pas pris en compte).

Si sadr n'est pas spécifié, toute la mémoire EEPROM sera effacée après demande de confirmation.

9. Organisation mémoire

La table mémoire du MC9S12D64 est initialisée par le moniteur TwinPEEKs comme suit (Note: en partie différente des valeurs par défaut du reset).

Begin	End	Ressource
\$0000	\$03FF	Registres de contrôle
\$0400	\$07FF	1KB EEPROM
\$3000	\$3FFF	4KB RAM (Défaut reset: \$0000-\$0FFF)
\$4000	\$7FFF	16KB Flash (équivalent à la Page \$3E)
\$8000	\$BFFF	16KB Flash page \$3C (n'importe quelle Page \$3C..\$3F, sélectionnable par PPAGE)
\$C000	\$FFFF	16KB Flash (équivalent à la Page \$3F)