計算機アーキテクチャ　80－10
（１９９０．１．２６）

マルチプロセッサシステムによるアセンブリ言語トレーニング

柳瀬龍郎　　秋山哲司
（福井大学工学部情報工学科）

アセンブリ言語によるプログラミング技術を初心者に教育する際に、出来るだけ計算機を裸に近い形で学習者に提示すると言う考え方に基ずき、マイクロプロセッサを使ったボードコンピュータを製作し実際に学習用に使用した。この時、おなじコンピュータを複数台作るなら、あとあと柔軟な使い方ができると考えてマルチマイクロコンピュータ（即ち、個々のＣＰＵが独立して制御できる非同期シリアル回線２組を各々が持つマルチマイクロプロセッサシステム）を製作しアセンブリ言語の学習用に使用した例について報告する。システムはつぎのような特徴をもっている。

①共有メモリ結合型

②モニタＲＯＭは共有メモリ空間内に持つこととし各プロセッサシステムはＲＡＭのみをローカル空間内に持つ。

③モニタ機能は、コンソールに関するＩ／Ｏ、別のＲＳ－２３２Ｃポートからのオブジェクトプログラムのダウンロード、ワンステップ機能など最小限のものとする。

Training for Assembly Language Programming on Multi-processor System

Tatsuro Yanase　　Tetsuji Akiyama

Department of Information, Faculty of Engineering, Fukui University

3-9-1 Bunkyou, Fukui-shi, Fukui 910, Japan

On the idea that it was werth while that the computer with nothing was presented to the students to study programming technique in first step, we made one bord computers , and used it practically.

Before making the system we thought that an multi micro computer (is a multi micro processor with two serial line) system should be appropriet for various use and the system was made. The system has following features:

①shared common memory

②the Rom only in a common memory space

③monitor has only few function, console I/O, down line loading, one step-trace, etc.

1　はじめに

著者らの所属する大学（福井大学工学部）では計算機を使ったプログラミング実習を行っているが、そのカリキュラムは以下のようになっている。

1）大型計算機を使ったＦＯＲＴＲＡＮのプログラミング実習　（１年生後期）
2）ワークステーションを使ったワンボードコンピュータでのアセンブリ言語のプログラミング実習　（２年生前期）
3）ＵＮＩＸワークステーションでのＣ言語のプログラミング実習　（２年生後期）
4）ＰＡＳＣＡＬによるコンパイラ作成実習　（３年生後期）

等、となっている。このうち、2）のアセンブリ言語のプログラミング実習では、大学スタッフの手作りによるワンボードコンピュータ（以後これをコンピュータエレメント、ＣＥと呼ぶ）を使っている。

ホストとなるワークステーションとシリアル回線で結合された端末の間にＣＥが入っている。ＣＥは２個のＲＳ－２３２Ｃポートを備えており簡単なモニタによってコントロールされている。そのモニタは、２つの回線から入力されたコードをそれぞれ他の回線へ送り出す機能を持っており、この機能を使うことにより端末は単純にホストＷＳのＴＳＳ端末として機能させることが出来る(端末機能)。

アセンブリ言語のプログラミング実習は次のステップによって進められる。

1）ホストＷＳが動作している状態でマルチコンピュータシステムの電源を投入したあと、ＣＥのリセットスイッチを押してモニタを起動する。
↓
2）モニタの端末機能を使ってＣＥをホストＷＳの端末として動作させ、ホストＷＳ上でアセンブリ言語のソースプログラムを作成する。この時ホストＷＳのＯＳであるＵＮＩＸの環境を使うことになる。
↓
3）ソースプログラムをＷＳ上でアセンブルしオブジェクトファイルを作成する。この時エラーがあればエディタに戻って修正を繰り返す。
↓
4）ＷＳのホストにＬＯＧＩＮしたままＣＥのモニタモードに戻ってホストからオブジェクトファイルをダウンロードする。
↓
5）ＣＥのＰＣレジスタその他必要なレジスタ類をセットしてプログラムを実行、またはデバッグする。この後必要が在れば１）～５）を繰り返す。
↓
6）ＣＥをホストＷＳの端末とし、ＬＯＧＯＵＴする。
↓
7）他にシステムを使っている人がいなければシステムの電源をＯＦＦにする

2　システムの構成

図１にシステム全体の構成図を示す。

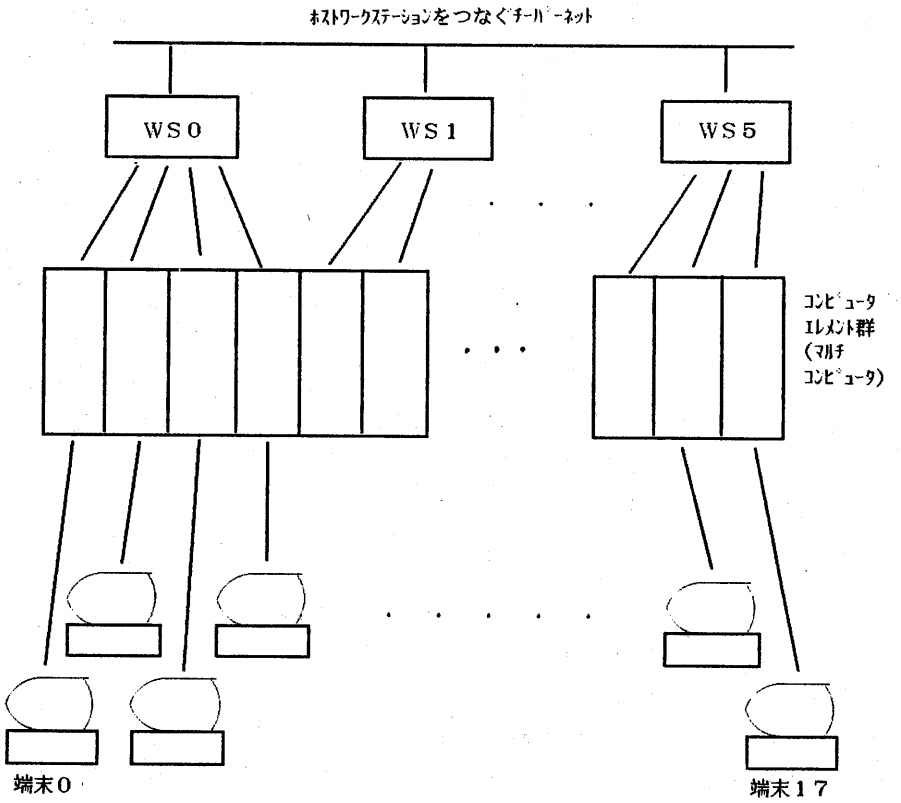
ホストＷＳはOMRON　ＬＵＮＡシリーズを５台使用し、ＯＳはＵＮＩＸシステムＶ系を用いている。ＣＥはＷＳと端末の間にあって、ＷＳからみればターゲットシステムであり、端末からみれば簡単なコマンドを受け付けるＲＯＭモニタを持つコンピュータとなっている。図２にＣＥの集りすなわちマルチコンピュータシステムのブロック図をしめす。ＣＥといっても実際は１枚の基版に３組のコンピュータが載っている。共通バスにはこれらのＣＥが結合されているだけでなく、どのＣＥからもアクセスされる共通ボードも結合されている。共通ボード上には、バスアービタ、モニタＲＯＭ，共通メモリなどが載っている。図３にＣＥのブロック図を示す。

3　ハードウエア

3・1　ＣＥボード

ＣＥは図３に示すように極めて単純であり、主として以下の要素で構成されている。

1）ＣＰＵ
モトローラ社　マイクロプロセッサMC６８００
８ 内部レジスタ　３２ｂｉｔ×１６　データバス



(端末は全体で２２台まで増設の予定)

図1　システムの全体図

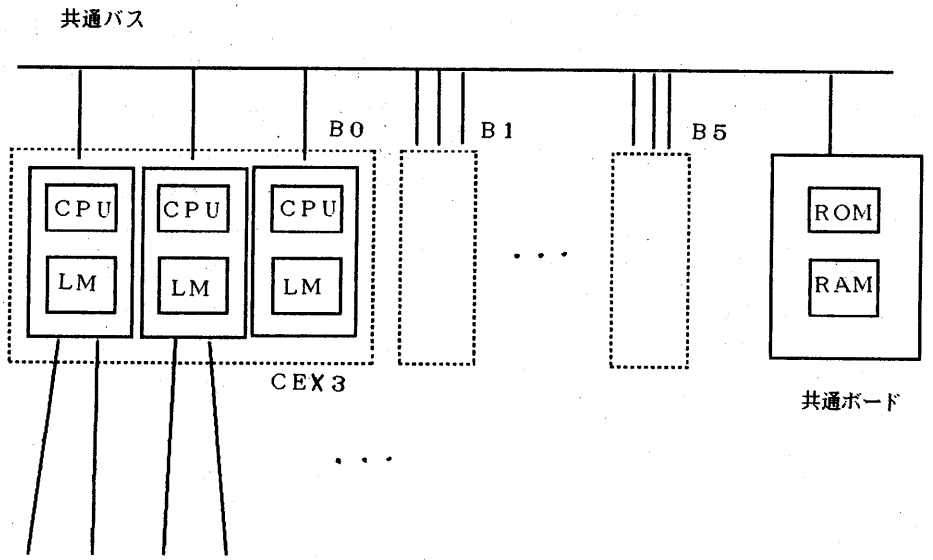


図2　マルチコンピュータのブロック図

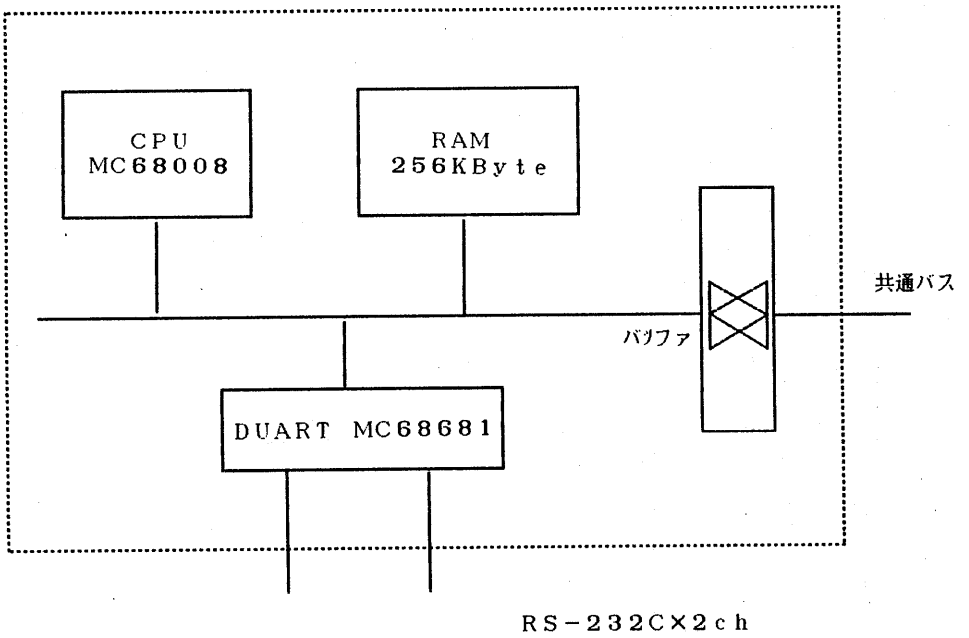


図3　CEのブロック図

8bit アドレスバス 20bit 8MHzを10MHzで使用。

2）メモリシステム

DRAMモジュール（MB85225－12）8bit×256kワード1個をDRAMC1422A 20MHzで制御。

3）回線制御

DUART MC68681PによりRS－232C×2chをサポート。

4）共通バスバッファ （アドレスバス16bit データバス8bit）。

3・2 共通ボード

以下のブロックから成る。

1）アービタ

20MHzのクロックによるシフトレジスタを使ったリングアービタ。

2）共通メモリ回路

（a）ROM MBM27C256A－20（32Kbyte）にCEのモニタが書き込まれている。

（b）RAM DRAMCにより256kword×8bitのDRAMモジュール（MB85225－12）を制御。

3・3 リセットメカニズム

個々のCEにはリセットスイッチが備えられており、CE毎にリセットが可能となっている。これは、学生のプログラムを走らせたとき暴走し、リセットする必要があっても他のCEに影響を与えないでリセットすることができるようにするためである。

CEのリセットボタンを押すことによって、デコーダーのハードウエアがリセットされ、CPUからの低位アドレスへのアクセスは共通ボード上のROMになされる。当然リセットベクタの取得においてもROMに向かってアクセスがなされる。その後、モニタのコピールーチンへジャンプする訳であるが、モニタのコピールーチンはROMの高位のアドレス部に格納されているため、CPUがここをアクセスした直後ハード的な処理がなされ、これによって以後のCPUの低

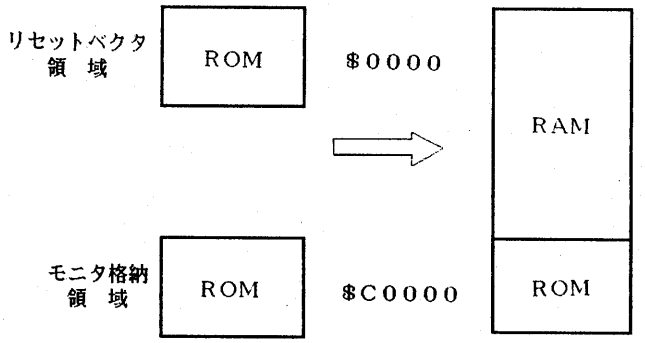


図4　メモリマップの変化

位アドレス部に対するアクセスは全てＲＯＭではなくＤＲＡＭに対してなされることになる。図４参照。
リセットベクタアクセス後のＣＰＵの仕事は、ＲＯＭ内に格納されているモニタをＲＯＭから、ＣＥ内のローカルＲＡＭへコピーし、モニタの先頭番地へジャンプすることである。これによってＣＥは端末からのコマンドを受け入れる状態となる。

4　メモリ空間

ＣＥのＣＰＵから見たメモリマップを次に示す。

①$0～$3000－1　　システム
②$3000～$3C000－1　ユーザ
③$3C000～$40000－1　システム
④$40000～$C0000－1 共通（ＤＲＡＭ）
⑤$C0000～$F0000－1 システム
　　（モニタ、ＲＯＭ）
⑥$F0000～$FFFFF　　システム
　　（Ｉ／Ｏその他）

Ｉ／Ｏポートアドレスは$Ｆ０００００である。また、回線ＬＳＩの内部レジスタマップは表１に示す。各ＣＥのローカルメモリ空間は自身のＣＰＵからのみアクセス可能であり、またそれとは逆に共通メモリ空間はどのＣＥからもアクセスが可能である。したがって、各ＣＥはこの共通ＲＡＭ領域を通じて、データ等の交換が可能である。

5　モニタ機能

5・1　モニタＲＯＭ

モニタは、$３Ｃ０００～$３ＦＦＦＦまでに格納されているものがリセット直後にＲＯＭからローカルＲＡＭコピーにされたものである。すなわち、全てのＣＥは、共通メモリ領域にあるただ一つのＲＯＭのモニタをコピーする訳である。したがって各ＣＥのモニタは全て同じ内容の物となる。このメカニズムのために個々のＣＥがおのおののＲＡＭを個別に持つ必要もなく、また改良したモニタの置き換えも同じ内容の多数のＲＯＭを用意する必要がないので容易に行うことができる。

5．2　モニタ機能

モニタは、１ステップトレース、メモリ内容表示・変更、レジスタ表示・変更、ブレークポイント設定・解除、実行、補助回線端末機能、Ｓレコードダウンロード機能、等のコマンドを受け付ける。ただし、これらのコマンド機能は随時改良変更している。モニタとは直接関係ないが、ＲＯＭには他の各種プログラムが格納されており、必要があればこれを随時使用するこ

表1　68681の内部レジスタ

RS4	RS3	RS2	RS1	Read (R/$\overline{W}$=1)		Write (R/$\overline{W}$=0)	
0	0	0	0	Mode Register A	(MR1A, MR2A)	Mode Register A	(MR1A, MR2A)
0	0	0	1	Status Register A	(SRA)	Clock-Select Register A	(CSRA)
0	0	1	0	Do Not Access*		Command Register A	(CRA)
0	0	1	1	Receiver Buffer A	(RBA)	Transmitter Buffer A	(TBA)
0	1	0	0	Input Port Change Register	(IPCR)	Auxiliary Control Register	(ACR)
0	1	0	1	Interrupt Status Register	(ISR)	Interrupt Mask Register	(IMR)
0	1	1	0	Counter Mode: Current MSB of Counter	(CUR)	Counter/Timer Upper Register	(CTUR)
0	1	1	1	Counter Mode: Current LSB of Counter	(CLR)	Counter/Timer Lower Register	(CTLR)
1	0	0	0	Mode Register B	(MR1B, MR2B)	Mode Register B	(MR1B, MR2B)
1	0	0	1	Status Register B	(SRB)	Clock-Select Register B	(CSRB)
1	0	1	0	Do Not Access*		Command Register B	(CRB)
1	0	1	1	Receiver Buffer B	(RBB)	Transmitter Buffer B	(TBB)
1	1	0	0	Interrupt-Vector Register	(IVR)	Interrupt-Vector Register	(IVR)
1	1	0	1	Input Port (Unlatched)		Output Port Configuration Register	(OPCR)
1	1	1	0	Start-Counter Command**		Output Port Register (OPR)	Bit Set Command**
1	1	1	1	Stop-Counter Command**			Bit Reset Command**

*This address location is used for factory testing of the DUART and should not be read. Reading this location will result in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.

**Address triggered commands.

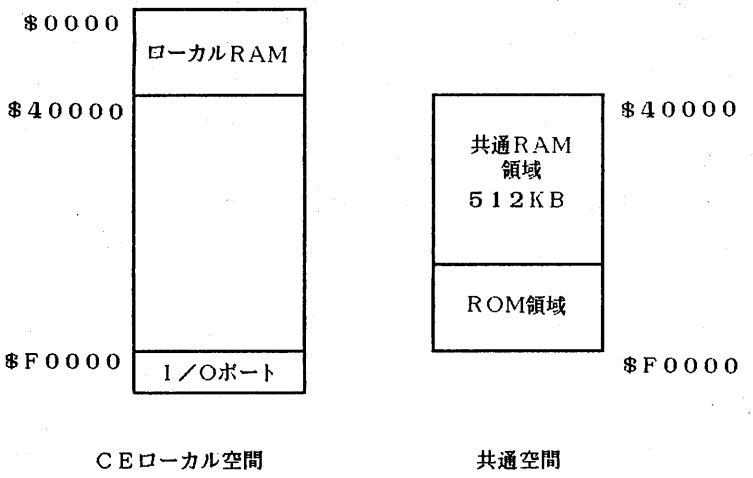


図5　メモリマップ

とができる。

6　使用形態

学生はホストコンピュータにアカウントを持っており、CEの端末モードでまずホストにログインする。このあとソースプログラムの作成とアセンブルを行ってオブジェクトのSフォーマットを作成する。これは、FORTRANで書かれたアセンブラを動かすことにより、ソースプログラム名、オブジェクトファイル名、アセンブルリスト名を順次聞いてくるので、それに答

えるだけでSフォーマットのオブジェクトファイルがホストのWS上に作られる。学生は担当教官から与えられた課題について、ソースプログラムをモトローラのアセンブリ言語ニーモニックを使って、ｖｉ等のＵＮＩＸ上のエディタで作成する。このあとエスケープコードを入力することにより、ＣＥのモニタモードにもどって、ホスト上のＳフォーマットのファイルをダウンロードするわけである。具体的にはコマンド文字の後にブランクをおいて、「ＣＡＴ　ｆｉｌｅ名」と入力すると、この「ＣＡＴ　ｆｉｌｅ名」をホストに送り出し、ホストから送られてくる文字をＳフォーマットに従ってメモリに格納する。

7　おわりに

本報告では市販のマイクロプロセッサを使ってボードレベルから自前で作り上げたシンプルなシステムをアセンブラプログラミングの教育に使った例について報告した。

初心者教育としてのアセンブラのプログラミング実習が一般的に必要かどうかは議論のあるところかも知れない。しかし、教育を行うとなれば、その言語の性質上、計算機の概念についてはできるだけ現実的に理解させることが必要と思われる。このような観点から我々は自前のシステムを用いて、初心者にアセンブラ言語教育を行っている。むろん、必ずしも効率の良い理想的な環境が得られた訳ではないが、このシステムを使うことによりすくなくとも学習者にとっては、システムの一部の資料がないとか、複雑すぎておおよそ理解の範囲をこえるといったような、市販のシステムを使った場合にありがちな欠点を負担として与えないですんだ。また学習者がシステム全体をシンプルではあるがコンピュータとして理解できる程度のものにできた。

システムを理解するときその一部でも不透明な部分があると全体が明確に理解できたという印象をもてないものである。また学習を終えたときに、対象の全体が把握できたと感じるなら、得られる満足感がきわめて大きいのも事実である。

今後、このマルチマイクロプロセッサを用いて、並行動作プログラミングによる、プログラミング技術を初心者教育の段階から学習させることを予定に今後の計画を進めたい。

==========

APPENDIX

MC68681のレジスタ

Channel A Command Register (CRA)

CRAは、Aチャンネルにコマンドを与えるのに用いられるレジスタである。コマンドが相反するものでない限り、CRAに一回の書き込みで複数のコマンドが指定できる。（すなわち、"enable transmitter"と"reset transmitter"は、一回のコマンドワードで指定できない。）

CRA[7]　このビットは、使用されてはおらず0あるいは1にセットされている。

CRA[6:4]　Aチャンネル　一般コマンド

0 0 0	No Command
0 0 1	Reset Mode Register Po
0 1 0	Reset Receiver
0 1 1	Resett Transmitter
1 0 0	Reset Error Status
1 0 1	Reset Channel A Break Change Interrupt
1 1 0	Start Break
1 1 1	Stop Break

CRA[3:2]　Aチャンネル　トランスミッタ　コマンド

0 0	No action is taken.
0 1	Enable Transmitter
1 0	Disable Transmitter
1 1	Do not use.

CRA[1:0]　Aチャンネル　レシーバー　コマンド

0 0	No action is taken.
0 1	Enable Receiver
1 0	Disable Receiver
1 1	Do not use.

Channel B Command Register (CRB)

このレジスタのビット定義は、(CRA)のビット定義と同様である。

Channel A Status Register (SRA)

SRA[7]　Aチャンネル　ブレーク　受信

このビットは、あらかじめ定められた長さの、ストップビットなしのオール0のキャラクターを受け取ったことを示す。このビットは、RxRDYビット(SRA[0]=1)がセットされた時にだけ有効となる。ブレークを受けとったFIFOバッファのみが満たされる。少なくとも半ビット以上のマーキング状態をシリアルラインから受け取るまでFIFOに対する入力は拒絶される。このビットをセットしよ

うとする時には、Aチャンネル割り込みステータスレジスタの" change in break" ビット(ISR[2])もまたセットされる。加えてISR[2]は、上で定義されたブレーク条件の最後を受け取ったときにセットされる。ブレーク検出回路は、受け取ったキャラクターの途中でブレークが始まるのを検出できる。

SRA[6] Aチャンネル フレーミングエラー

このビットが、セットの時は、FIFOの一致しているデータキャラクターが受け取られた時にストップビットが検出されなかったことを示している。ストップビットのチエックは、最初のストップビットの中央の位置でチェックされる。このビットは、RxRDYビット(SRA[0]=1)がセットされた時にだけ有効となる。

SRA[5] Aチャンネル パリティーエラー

"with parity" あるいは " force parity" モードが、モードレジスタ1(MR1A)によってプログラムされ、FIFO中の対応しているキャラクターが間違ったパリティーとともに受け取られる。マルチドロップモードでは、受け取ったアドレス/データビットをストアするために,パリティーエラービット条件は、使われる。このビットは、RxRDYビット(SRA[0]=1)がセットされた時にだけ有効となる。

SRA[4] Aチャンネル オーバーランエラー

このビットが、セットの時は、受け取ったデータストリームの中の一つあるいは、それ以上のキャラクターが、失われてしまったことを示している。このビットは、FIFOがいっぱいで、レシーブシフトレジスタのキャラクターがすでに空のFIFO条件を待っている時に、セットされるようになる。これが起きると、レシーブシフトレジスタのキャラクターは、(もしあれば、そのブレーク検出、パリティーエラー、そしてフレーミングステータス) 失われる。このビットは、リセットエラーステータスコマンドでクリアされる。

SRA[3] Aチャンネル トランスミッタ エンプティー

このビットは、トランスミッタがアンダーランの時セットされるようになる。すなわち、トランスミットホールディングレジスタとトランスミットシフトレジスタの両方が、空である。もし、トランスミットホールディングレジスタのキャラクターが伝送を待っていなければ、キャラクターの最後のストップビットの伝送後にこのビットは、セットされる。トランスミットホールディングレジスタがCPUによってロードされるか、あるいはトランスミッタがディスエーブルされる時にこのビットはクリアされる。

SRA[2] Aチャンネル トランスミッタ レディー

このビットが、セットの時は、トランスミットホールディングレジスタが空であり、キャラクターをロードするための準備をしていることを示している。トランスミッタレディーは、トランスミットシフトレジスタにキャラクターが伝送される時にセットされる。トランスミッタレディーは、また、トランスミッタがディスエーブルされる時クリアされ、そしてトランスミッタが最初にイネーブルされる時にセットされる。すなわち、トランスミッタがディスエーブルされている間は、トランスミッタホールディングレジスタにロードされたキャラクターは、伝送されない。

SRA[1] Aチャンネル FIFO FULL

このビットは、キャラクターがレシーブシフトレジスタからレシーバーFIFOに伝送されるときにセットされ、そしてその伝送がFIFOをいっぱいにさせる。すなわち、三つのFIFOホールディングレジスタ全てが占有される。このビットは、CPUがレシーバーバッファを読み込むときにクリアされる。FIFOがいっぱいのために、もしキャラクターがレシーブシフトレジスタで待っているならば、AチャンネルFIFOfullステータスビットは、CPUがレシーバーバッファを読み込むときにクリアされない。

SRA[0] Aチャンネル レシーバー レディー

このビットは、キャラクターはすでに受け取られて、CPUによって読み込まれるためにFIFOで待っていることを示す。このビットは、キャラクターがレシーブシフトレジスタからレシーバーFIFOに伝送されるときにセットされ、CPUがレシーバーバッファを読み込むときにクリアされる。

Channel B Status Register (SRB)

このレジスタのビット定義は、SRA)のビット定義と同様じである。