第6章 MicroPythonでビット・バンギング

I²C & SPIをゼロから作る⑤ …SPI送受信機能の実装

岡野 彰文

ステップ1…MicroPythonで基本実装部分を作る

それではMicroPythonでビット・バンギングによるメイン側の実装を見てみましょう．I²Cに比べると単純です．

クラス内にはイニシャライザ__init__()と，転送を行うwrite_readinto()メソッドだけが定義されています．

SPIでは送受信を同時に行うので，転送を行うAPIにはライト，リードの区別はありません．

イニシャライザではGPIOのインスタンスを受け取ってそれを保持，初期化を行い，さらに転送ビット数とMSB/LSBファーストの指定により転送ビット順を決めるようにしてあります．

write_readinto()メソッドは送信と受信用のバッファ(リスト)を渡して通信を行います．

● Picoのピン入出力設定

SPIのMISOピンは入力に固定，SCLKとCSピンは出力に固定したままとなります．これは転送中に切り替わることはありません．

MOSIピンはデータ出力ピンなので出力設定が基本ですが，非転送時には入力に切り替えることによってハイ・インピーダンス状態にしています．ただしこれは，他のデバイスとの互換性を考える必要がなければ必須の機能ではありません．

● 初期設定

bbSPIのイニシャライザには幾つかのキーワード引数が与えられます．これらのキーワードはmachine.SPIに倣っています．

ピンの指定はsck，mosi，misoとcsのとしてGPIOピンのインスタンスを渡します．

CSピンを設定するかどうかはオプションです．CSピンを指定しておくと，転送の際に自動的にCSピンの制御が行われます．指定がなかった場合はこの制御は行われないので，SPI通信を行うたびにその前後でCSピンのアサート/デアサートを行わなくてはなりません．

もし同じbbSPIインスタンスで，並列に接続したサブノードに対し個別のチップ・セレクト信号を用いる場合は，このCSの設定は行わず，write_readinto()メソッドをコールする前後でそれぞれのピンを制御してください．

ところでmachine.SPIクラスにはbaudrate設定があり，SCLKのクロック周波数を決めることができます．しかしbbSPIには用意されていません．bbSPIは，bbI2Cと同じようにクロック周波数が低いため，あえてこの設定を設けていません．

porarity，phaseでモード設定を行います．この名称はMicroPythonのSPIクラスに準じたもので，polarity，phaseはそれぞれCPOL，CPHAの設定です．

さらにビット順first_bitとビット長bitsの設定があります．first_bitにはMSB(＝1)またはLSB(＝0)を指定できます．ビット長には制限はありません．転送データは整数のリストとして用意することを想定しています．MicroPythonのベースとなっているPyhton3では整数のサイズに上限はないため，これを利用して長いビット長にも対応できるようにしてあります(machine.SPIクラスと同様)．

初期設定では上記の設定の保存に加えて，SCLKとCS信号の初期化も行っています．

● データ送受信

転送を行うwrite_readinto()メソッド内ではCSをアサートし，クロックをトグルさせ，データをビットごとに送受信します．

ステップ2…MicroPythonでユーザ向けのAPIを用意する

● machine.SPIに倣った簡略化APIとした

コンストラクタのためのインターフェースは，machine.SPIに倣ったキーワード引数を設けました．