一般社団法人 組込みシステム技術協会　技術本部 技術高度化委員会

状態遷移設計研究会

ソースコードから状態遷移表のリバースツール

RExSTM for C

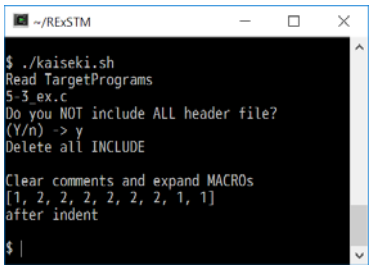
目的

- 状態遷移表の普及・定着のための、プロセス研究。
- ソフトウェアの 品質向上、開発効率改善！
- 今まで検証したリバース手順をツールで効率的に実現

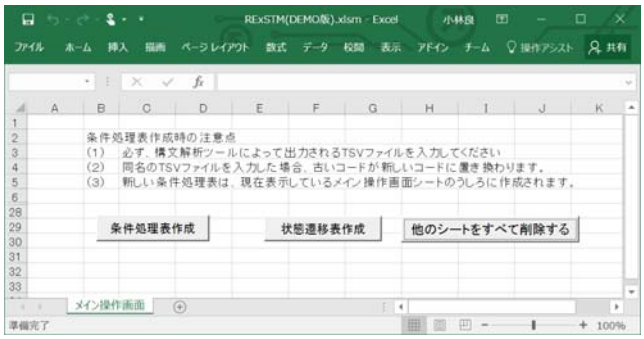
機能

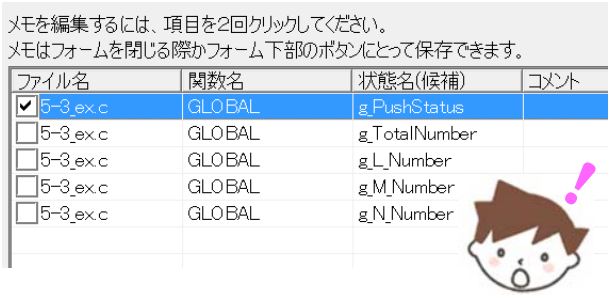
Step.1 レガシーコードを解析にかけて、

```
extern    int    in1,in2
int       state;
#define S1  1
#define S2  2
#define S3  3
#define X1 5
#define X2 T1*2
int       out

void task(void)
{
        int       t,x     ;

        t = in1;
        s = in2;
        if ( t == 1) {
                s++;
                if (s < X1) {
                        if(  state == S1 ){
                                out = s;
                        else if( state == S2 ){
                                out = 0;
                                state = S3
```



Step.2 ボタン１つで状態変数候補を自動リストアップ！





Step.3 状態変数を選択すれば、状態遷移表を自動作成！

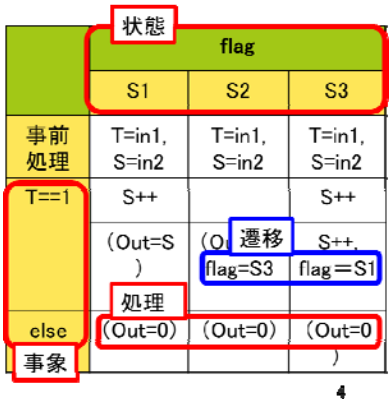
		g_pushStatus						
		0	1	2	3	4	5	6
void ENTRY_Calculate	(g_L_Number == g_M_Number) && (g_M_Number == g_R_Number)	–	–	–	g_pushStatus = 4; g_TotalNumber += ((g_BetNumber << 2) + g_BetNumber);	–	–	–
	(g_L_Number == g_M_Number) || (g_L_Number == g_R_Number) || (g_M_Number == g_R_Number)	–	–	–	g_pushStatus = 5; g_TotalNumber += g_BetNumber;	–	–	–
	ELSE	–	–	–	g_pushStatus = 6;	–	–	–
void ENTRY_ChangeStatus	無条件	g_pushStatus = 1;	g_pushStatus = 2;	g_pushStatus = 3;	–	g_pushStatus = 0;	g_pushStatus = 0;	g_pushStatus = 0;

成果

- ソースコードから状態遷移表生成手順の多くを自動化！
- 状態変数の候補を自動抽出、選ぶだけでOK！

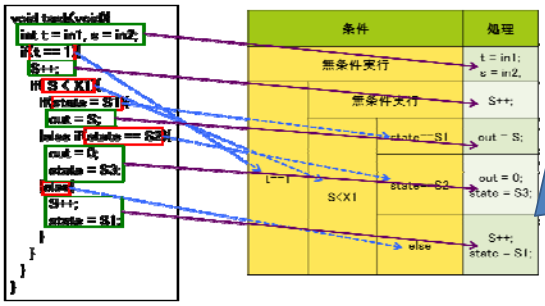
詳細は配布資料・プレゼンにてご紹介いたします。

- 研究の背景と目的
  - 組込みソフトウェア開発の傾向：再利用性や保守性向上の要求がある
    - 組込みシステムの高度化による開発の大規模化，複雑化
    - 高品質・低コスト・短納期によるソフトウェア開発の要求
    - 抜け・漏れを検出したい
  - ソースコードのレガシー化によって，再利用や保守におけるコストが増大している現状がある．
  - → レガシーコードを容易に理解したい．



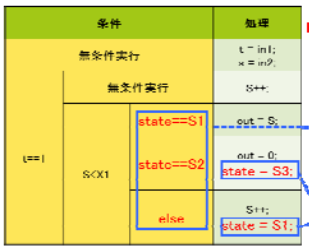
レガシーコードを解析し，仕様の理解を支援する
→ レガシーコードから状態遷移表を抽出する

- 条件処理表の作成手順



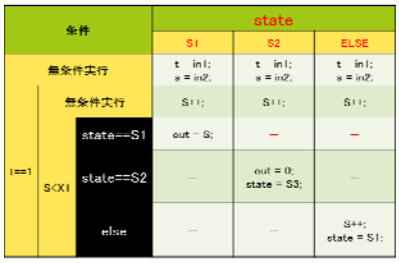
条件処理表：
左列に条件，右列に対応する処理を記述した表である．条件は，プログラム中のif文やswitch文における条件式である．

条件処理表（仮状態遷移表）から状態変数の選択



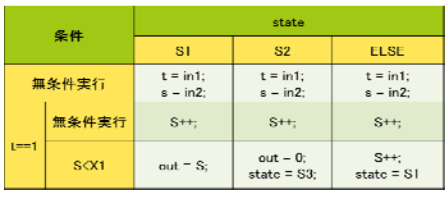
状態変数候補：
左の3つの条件をすべて満たしている変数で，複数存在しうる．
状態変数として，1つだけ選択されるものとする．

選択した状態変数に対する状態遷移表の作成



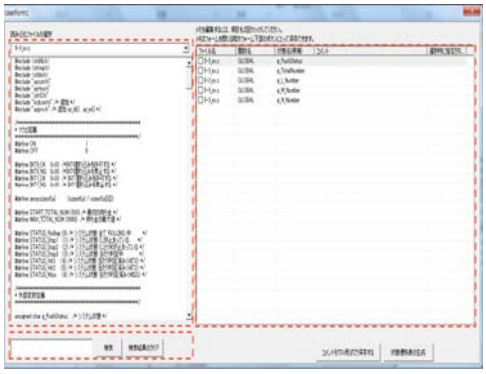
状態遷移表（未完）：
条件処理表の処理列を複製して，存在しない処理をハイフンに置き換えた状態の表．
列数は，状態変数が取りうる状態の数である．

不要な記述の削除



状態遷移表：
上の状態遷移表を整理したもの．
無条件実行とは，あるイベントを実行する前に必ず実行される，事前処理を表す．
または，事後処理のある場合もある．

- ツール化
  - 左の手法をツール化する
  - Python＋Excelで実装する
    - Python：構文解析
    - Excel VBA：各表の表示
  - 状態変数選択のためのフォーム



  - 手動による手法適用と比較した
    - 対象ソースコード1：150ステップ程度
    - 対象ソースコード2：400ステップ程度
  - 手動だと間違いがあった
    - プログラムの解釈を間違える
    - 表を作成する際に，位置がおかしくなるなどの作成上の間違いがあった
    - 状態変数を探すのに時間がかかる上，選択上のミスもあった
  - 手動では，多くの時間が必要であった
    - 作成者によるが，1時間以上は必要であった
    - 修正・レビューを繰り返す必要があった

手動による手法適用にくらべて，
- **間違いがない**
- **10%以下の時間で作成できる**