ファイル名	関数名	解説	対応有無	理由	備考			
Splint 3.1.1 --- 12 April 2003								
Drv¥DrvPort.c(21,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFC0).DDR = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。	To make char and int types equivalent, use +charint.			
Drv¥DrvPort.c(22,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFC0).DR.BYTE = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(25,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p2 *)0xFFFFC1).DDR = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(26,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p2 *)0xFFFFC1).DR.BYTE = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(27,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p2 *)0xFFFFC1).PCR.BYTE = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(30,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFC4).DDR = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(31,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFC4).DR.BYTE = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(34,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p4 *)0xFFFFC5).DDR = 0x0f	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(35,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p4 *)0xFFFFC5).DR.BYTE = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(37,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p4 *)0xFFFFC5).PCR.BIT.B4 = 1	×	ビットフィールドの範囲内の設定は許容する。				
Drv¥DrvPort.c(38,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p4 *)0xFFFFC5).PCR.BIT.B5 = 1	×	ビットフィールドの範囲内の設定は許容する。				
Drv¥DrvPort.c(39,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p4 *)0xFFFFC5).PCR.BIT.B6 = 1	×	ビットフィールドの範囲内の設定は許容する。				
Drv¥DrvPort.c(40,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p4 *)0xFFFFC5).PCR.BIT.B7 = 1	×	ビットフィールドの範囲内の設定は許容する。				
Drv¥DrvPort.c(43,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p5 *)0xFFFFC8).DDR = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(44,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p5 *)0xFFFFC8).DR.BYTE = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(45,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p5 *)0xFFFFC8).DR.BIT.B0 = 0	×	ビットフィールドの範囲内の設定は許容する。	Types are incompatible. (Use -type to inhibit warning)			
Drv¥DrvPort.c(46,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p5 *)0xFFFFC8).DR.BIT.B1 = 0	×	ビットフィールドの範囲内の設定は許容する。				
Drv¥DrvPort.c(49,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p6 *)0xFFFFC9).DDR = 0xff	×	ビットフィールドの範囲内の設定は許容する。				
Drv¥DrvPort.c(50,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p6 *)0xFFFFC9).DR.BYTE = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(56,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p8 *)0xFFFFCD).DDR = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(57,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p8 *)0xFFFFCD).DR.BYTE = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(60,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p9 *)0xFFFFD0).DDR = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(61,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p9 *)0xFFFFD0).DR.BYTE = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(64,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFD1).DDR = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(65,2):	InitPort	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFD1).DR.BYTE = 0xff	×	型の違いだが16進表記であり範囲を超えていないことを確認した。				
Drv¥DrvPort.c(78,2):	LedOnB0	Assignment of int to unsigned char: (*(volatile struct st_p5 *)0xFFFFC8).DR.BIT.B0 = 1	×	ビットフィールドの範囲内の設定は許容する。				
Drv¥DrvPort.c(86,2):	LedOffB0	Assignment of int to unsigned char: (*(volatile struct st_p5 *)0xFFFFC8).DR.BIT.B0 = 0	×	ビットフィールドの範囲内の設定は許容する。				
Drv¥DrvPort.c(94,2):	LedOnB1	Assignment of int to unsigned char: (*(volatile struct st_p5 *)0xFFFFC8).DR.BIT.B1 = 1	×	ビットフィールドの範囲内の設定は許容する。				
Drv¥DrvPort.c(102,2):	LedOffB1	Assignment of int to unsigned char: (*(volatile struct st_p5 *)0xFFFFC8).DR.BIT.B1 = 0	×	ビットフィールドの範囲内の設定は許容する。				
Drv¥DrvSdCard.c(163,2):		Initial value of writeStatus.fillData is type int, expects	×	定数であり範囲内なので問題ない				

ファイル名	関数名	解説	対応有無	理由	備考			
Drv¥DrvSdCard.c(286,2):	ResetCard	Return value (type unsigned short int) ignored: SendSdRepeat((ch...	×	SendSdRepeat()の戻り値は受信したデータ長だがここでは必要ないため参照しない。	Result returned by function call is not used. If this is intended, can cast result to (void) to eliminate message. (Use -retvalint to inhibit warning)			
Drv¥DrvSdCard.c(290,14):	ResetCard	Function publishCmd expects arg 1 to be char gets int: (0)	×	定数であり範囲内であるため明示的なキャストはおこなわない。				
Drv¥DrvSdCard.c(290,2):	ResetCard	Return value (type unsigned short int) ignored: publishCmd((0), 0L)	×	コマンド発行でエラーを見ていないが次の工程でレスポンスが返ってこないので異常は検出できるため判定は無くてもよい。				
Drv¥DrvSdCard.c(291,25):	ResetCard	Function waitForReply expects arg 1 to be unsigned char gets int: 0x0ff	×	定数であり常に範囲内であることを確認したので問題はない。				
Drv¥DrvSdCard.c(291,32):	ResetCard	Function waitForReply expects arg 2 to be unsigned char gets int: (0x01)	×	定数であり常に範囲内であることを確認したので問題はない。				
Drv¥DrvSdCard.c(301,4):	ResetCard	Return value (type unsigned short int) ignored: SendSdOne('¥xff')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(304,16):	ResetCard	Function publishCmd expects arg 1 to be char gets int: (1)	×	定数であり常に範囲内であることを確認したので問題はない。				
Drv¥DrvSdCard.c(304,4):	ResetCard	Return value (type unsigned short int) ignored: publishCmd((1), 0L)	×	コマンド発行でエラーを見ていないが次の工程でレスポンスが返ってこないので異常は検出できるため判定は無くてもよい。				
Drv¥DrvSdCard.c(305,27):	ResetCard	Function waitForReply expects arg 1 to be unsigned char gets int: 0xff	×	定数であり常に範囲内であることを確認したので問題はない。				
Drv¥DrvSdCard.c(305,33):	ResetCard	Function waitForReply expects arg 2 to be unsigned char gets int: 0x00	×	定数であり常に範囲内であることを確認したので問題はない。				
Drv¥DrvSdCard.c(316,2):	ResetCard	Return value (type unsigned short int) ignored: SendSdOne('¥xff')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(392,3):	ReadSdPhysical	Implicitly only storage readStatus.buff (type char *) not released before assignment: readStatus.buff = buff	×	システム全般に動的なメモリアロケーションは使用していない。また、直前に前回の使用が終了したかを判定しているので問題はない。	A memory leak has been detected. Only-qualified storage is not released before the last reference to it is lost. (Use -mustfreeonly to inhibit warning)			
Drv¥DrvSdCard.c(392,3):1	ReadSdPhysical	Implicitly temp storage buff assigned to implicitly only: readStatus.buff = buff	×	バッファのポインタを置き換えるが元の内容がすでに使われていないことは直前で確認されている。	Temp storage (associated with a formal parameter) is transferred to a non-temporary reference. The storage may be released or new aliases created. (Use -temptrans to inhibit warning)			
Drv¥DrvSdCard.c(398,2):	ReadSdPhysical	Variable buff is kept in false branch, but not kept in true branch.	×	true の分岐はエラーのケースなので buff を保存しなくても問題はない。	The state of a variable is different depending on which branch is taken. This means no annotation can sensibly be applied to the storage. (Use -branchstate to inhibit warning)	Drv¥DrvSdCard.c(398,2): in false branch:	Drv¥DrvSdCard.c(392,3): Storage buff becomes kept	
Drv¥DrvSdCard.c(449,17):	readSdControl	Function kickCmd expects arg 1 to be char gets int: (17)	×	kickCmdの第一引数はSDコマンドでcharのサイズで指定することとが明白であり範囲内の定数なので問題はない。				
Drv¥DrvSdCard.c(473,9):	readSdControl	Operands of == have incompatible types (char, int): (*buffSrc & 0x80) == 0x00	×	型の違いが影響を受けないようにビット位置のマスクをしたのちに判定をしているので問題ない。				
Drv¥DrvSdCard.c(496,6):	readSdControl	Assignment of int to unsigned short int: readStatus.result = (0x0030) | (0x0009)	×	エラーコードはそれとわかる名称でマクロ定義されており範囲外の値の入ることの無いように定義されている。この部分ではそのマクロ定義を使用しており問題はない。	To ignore signs in type comparisons use +ignoresigns			
Drv¥DrvSdCard.c(504,5):	readSdControl	Assignment of int to unsigned short int: readStatus.result = BitToNum((unsigned short int)response) + 1	×	元のresponseは 8bit のデータなので BitToNum() の戻り値は0～7の範囲である。+1でunisgned short の範囲を超えることはない。				
Drv¥DrvSdCard.c(516,8):	readSdControl	Fall through case (no preceding break)	×	breakしない旨コメントにも明記してある。割り込みの発生しないケースなので継続して次のcaseブロックを実行する。	Execution falls through from the previous case. (Use -casebreak to inhibit warning)			
Drv¥DrvSdCard.c(522,9):	readSdControl	Operands of == have incompatible types (char, int): (*buffSrc & 0xff) == (0xfe)	×	charのデータであることを意識して0xffでマスクしてから比較しているので問題ない。				
Drv¥DrvSdCard.c(525,6):	readSdControl	Assignment of int to unsigned short int: dataPoint = i + 1	×	左右のデータはどちらもunsigned short で定数の「1」を加えている。データの範囲を意識していて問題もない。				

ファイル名	関数名	解説	対応有無	理由	備考			
Drv¥DrvSdCard.c(579,8):	readSdControl	Fall through case (no preceding break)	×	breakしない旨コメントにも明記してある。割り込みの発生しないケースなので継続して次のcaseブロックを実行する。				
Drv¥DrvSdCard.c(595,8):	readSdControl	Fall through case (no preceding break)	×	breakしない旨コメントにも明記してある。割り込みの発生しないケースなので継続して次のcaseブロックを実行する。				
Drv¥DrvSdCard.c(611,8):	readSdControl	Fall through case (no preceding break)	×	breakしない旨コメントにも明記してある。割り込みの発生しないケースなので継続して次のcaseブロックを実行する。				
Drv¥DrvSdCard.c(613,4):	readSdControl	Return value (type unsigned short int) ignored: SendSdOne('¥xff')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(614,4):	readSdControl	Return value (type unsigned short int) ignored: SendSdOne('¥xff')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(617,8):	readSdControl	Fall through case (no preceding break)	×	breakしない旨コメントにも明記してある。割り込みの発生しないケースなので継続して次のcaseブロックを実行する。				
Drv¥DrvSdCard.c(619,4):	readSdControl	Return value (type unsigned short int) ignored: SendSdOne('¥xff')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(886,3):	writeBlock	Implicitly only storage writeStatus.buff (type char *) not released before assignment: writeStatus.buff = writeData	×	動的なメモリ確保は一切行っていない上に直前でこの領域の使用状況をチェックしているのでメモリ解放の問題は無い。				
Drv¥DrvSdCard.c(886,3):1	writeBlock	Implicitly temp storage writeData assigned to implicitly only: writeStatus.buff = writeData	×	writeBlock()に引き渡されるwriteData変数はこの関数が抜けた後も使われるルールであり一時メモリを使い続けるという問題は発生しない。				
Drv¥DrvSdCard.c(890,2):	writeBlock	Variable writeData is kept in false branch, but not kept in true branch.	×	true の分岐はエラーのケースなので writeData を保存しなくても問題はない。	Drv¥DrvSdCard.c(890,2): in false branch:	Drv¥DrvSdCard.c(886,3): Storage writeData becomes		
Drv¥DrvSdCard.c(926,17):	writeSdControl	Function kickCmd expects arg 1 to be char gets int: (24)	×	定数であり常に範囲内であることは明確なので問題はない。				
Drv¥DrvSdCard.c(936,39):	writeSdControl	Function RecvSdKick expects arg 3 to be unsigned short int gets unsigned long int: (8UL)	×	unsigned long として計算する場合があるのでULを付加しているがunsigned shortの範囲に入ることは明確な定数なので問題	To ignore type qualifiers in type comparisons use +ignorequals.			
Drv¥DrvSdCard.c(946,16):	writeSdControl	Operands of < have incompatible types (unsigned short int, unsigned long int): i < (8UL)	×	unsigned short に入ることが明確な定数であり、符号無しどうしの比較なので問題は				
Drv¥DrvSdCard.c(948,9):	writeSdControl	Operands of == have incompatible types (char, int): transBuff[i] == 0x00	×	明示的な定数との比較で型が問題となる値でもない				
Drv¥DrvSdCard.c(958,5):	writeSdControl	Assignment of int to unsigned short int: writeStatus.result = (0x0020) | (0x0009)	×	代入している定数は代入先の変数のサイズを意識しており問題はない				
Drv¥DrvSdCard.c(965,5):	writeSdControl	Assignment of int to unsigned short int: writeStatus.result = BitToNum((unsigned short int)response) + 1	×	元のresponseは 8bit のデータなので BitToNum() の戻り値は0～7の範囲である。+1でunisgned short の範囲を超えることはない。				
Drv¥DrvSdCard.c(972,5):	writeSdControl	Return value (type unsigned short int) ignored: SendSdOne('¥xff')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(973,5):	writeSdControl	Return value (type unsigned short int) ignored: SendSdOne((char)...	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(999,4):	writeSdControl	Return value (type unsigned short int) ignored: SendSdOne('¥0')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(1000,4):	writeSdControl	Return value (type unsigned short int) ignored: SendSdOne('¥0')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(1018,9):	writeSdControl	Operands of == have incompatible types (char, int): (transBuff[i] & (0x11)) == (0x01)	×	一部のビットを比較するためにマスクしている。そのため8ビットを超えることはなく問題とならない。				
Drv¥DrvSdCard.c(1034,5):	writeSdControl	Assignment of int to unsigned short int: writeStatus.busySearchPoint = i + 1	×	unsigned shortの変数に足している数は明示されている定数であり変数のサイズを超えることは無いので問題はない。				
Drv¥DrvSdCard.c(1066,8):	writeSdControl	Fall through case (no preceding break)	×	割り込みが起きない場合はそのまま次のステージを実行するのでbreakはおこなわない。breakを意識しておこなっていないことをコメントで明示している。				

ファイル名	関数名	解説	対応有無	理由	備考			
Drv¥DrvSdCard.c(1072,5):	writeSdControl	Return value (type unsigned short int) ignored: ChangeSdRate(syn...	×	エラー時の回復処理なのでこのエラーの戻り値をセットすると元のエラーが何か分からなくなってしまうのでここでは見ない。				
Drv¥DrvSdCard.c(1078,9):	writeSdControl	Operands of == have incompatible types (char, int): (transBuff[i] & 0x0ff) == 0x0ff	×	マスクしているので型の違いの問題はない				
Drv¥DrvSdCard.c(1102,8):	writeSdControl	Fall through case (no preceding break)	×	割り込みが起きない場合はそのまま次のステージを実行するのでbreakはおこなわない。breakを意識しておこなっていないことをコメントで明示している。				
Drv¥DrvSdCard.c(1105,4):	writeSdControl	Return value (type unsigned short int) ignored: SendSdOne('¥xff')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(1229,31):	ReadAndCheckCs	Function GetR1ResponseData expects arg 1 to be char gets int: (9)	×	定数であり代入先の型に収まることは明確なので問題ない。				
Drv¥DrvSdCard.c(1263,18):	changeBlockSize	Function publishCmd expects arg 1 to be char gets int: (16)	×	定数であり代入先の型に収まることは明確なので問題ない。				
Drv¥DrvSdCard.c(1265,28):	changeBlockSize	Function waitForReply expects arg 1 to be unsigned char gets int: (0x80)	×	定数であり代入先の型に収まることを確認した。				
Drv¥DrvSdCard.c(1265,36):	changeBlockSize	Function waitForReply expects arg 2 to be unsigned char gets int: 0x00	×	定数であり代入先の型に収まることを確認した。				
Drv¥DrvSdCard.c(1269,2):	changeBlockSize	Return value (type unsigned short int) ignored: SendSdOne('¥xff')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdCard.c(1282,3):	changeBlockSize	Assignment of int to unsigned short int: err = BitToNum(response) + 1	×	元のresponseは 8bit のデータなので BitToNum() の戻り値は0～7の範囲である。+1でunisgned short の範囲を超えることはない。				
Drv¥DrvSdCard.c(1305,6):	CheckCsd	Operands of != have incompatible types (unsigned char, int): csd.field.always1 != 1	×	定数との比較で比較元の型に収まることは明確なので問題は無い。				
Drv¥DrvSdCard.c(1404,12):	CalcReadTimeout	Array fetch using non-integer, unsigned char: TaacExpTable[csd.field.taacUnit]	×	インデックスは符号なしの3ビットのビットフィールドであり元の配列の数を超えることが無いことを確認した。	To allow char types to index arrays, use +charindex. (Use +charindex to inhibit warning)			
Drv¥DrvSdCard.c(1407,12):	CalcReadTimeout	Array fetch using non-integer, unsigned char: TaacManTable[csd.field.taacValue]	×	インデックスは符号なしの4ビットのビットフィールドであり元の配列の数を超えることが無いことを確認した。				
Drv¥DrvSdCard.c(1437,6):	CalcReadTimeout	Operands of != have incompatible types (unsigned char, int): csd.field.nsac != 0	×	定数との比較であり比較元の型を超えないことが明確なので問題は無い。				
Drv¥DrvSdCard.c(1506,6):	CalcRate	Operands of <= have incompatible types (unsigned char, int): csd.field.rateUnit <= 3	×	定数との比較であり比較元の型を超えないことが明確なので問題は無い。				
Drv¥DrvSdCard.c(1508,13):	CalcRate	Array fetch using non-integer, unsigned char: rateExpTable[csd.field.rateUnit]	×	インデックスは符号なしの3ビットのビットフィールドであり元の配列の数を超えることが無いことを確認したので問題は無い。				
Drv¥DrvSdCard.c(1512,12):	CalcRate	Array fetch using non-integer, unsigned char: rateManTable[csd.field.rateValue]	×	インデックスは符号なしの4ビットのビットフィールドであり元の配列の数を超えることが無いことを確認したので問題は無い。				
Drv¥DrvSdCard.c(1629,24):	publishCmd	Incompatible types for + (int, char): 0x40 + cmd	×	intの0x40は定数であり加算対象の型を超えないことを確認したので問題は無い。				
Drv¥DrvSdCard.c(1634,2):	publishCmd	Assignment of int to char: transBuff[5] = 0x95	×	代入している定数は代入先の変数のサイズを意識しており超えないことを確認したので問題はない				
Drv¥DrvSdCard.c(1660,24):	kickCmd	Incompatible types for + (int, char): 0x40 + cmd	×	intの0x40は定数であり加算対象の型を超えないことを確認したので問題は無い。				
Drv¥DrvSdCard.c(1665,2):	kickCmd	Assignment of int to char: transBuff[5] = 0x95	×	代入している定数は代入先の変数のサイズを意識しており超えないことを確認したので問題はない				
Drv¥DrvSdCard.c(1731,28):	GetR1ResponseD	Function waitForReply expects arg 1 to be unsigned char gets int: (0x80)	×	定数であり代入先の型に収まることを確認した。				
Drv¥DrvSdCard.c(1731,36):	GetR1ResponseD	Function waitForReply expects arg 2 to be unsigned char gets int: 0x00	×	定数であり代入先の型に収まることを確認した。				
Drv¥DrvSdCard.c(1735,28):	GetR1ResponseD	Function waitForReply expects arg 1 to be unsigned char gets int: 0xff	×	定数であり代入先の型に収まることを確認した。				
Drv¥DrvSdCard.c(1735,34):	GetR1ResponseD	Function waitForReply expects arg 2 to be unsigned char gets int: (0xfe)	×	定数であり代入先の型に収まることを確認した。				
Drv¥DrvSdCard.c(1748,5):	GetR1ResponseD	Return value (type unsigned short int) ignored: SendSdRepeat((ch...	×	戻り値は受信したデータ長だがここでは使用しないので戻り値は見ない。				

ファイル名	関数名	解説	対応有無	理由	備考			
Drv¥DrvSdCard.c(1758,3):	GetR1ResponseD	Assignment of int to unsigned short int: retCode = BitToNum((unsigned short int)response) + 1	×	元のresponseは 8bit のデータなので BitToNum() の戻り値は0～7の範囲である。+1でunisgned short の範囲を超えることはない。				
Drv¥DrvSdCard.c(1762,2):	GetR1ResponseD	Return value (type unsigned short int) ignored: SendSdOne('¥xff')	×	戻り値は受信したデータだがここでは使用しないため見る必要はない。				
Drv¥DrvSdDmaSciItu.c(126,2):	InitSdTrans	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.TE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(127,2):	InitSdTrans	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(128,2):	InitSdTrans	Assignment of int to unsigned char: (*(volatile union un_scmr *)0xFFFFB6).BIT.SDIR = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(129,2):	InitSdTrans	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SMR.BIT.CA = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(133,2):	InitSdTrans	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.TE = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(134,2):	InitSdTrans	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RE = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(162,2):	SendSdOne	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RIE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(165,7):	SendSdOne	Operands of == have incompatible types (unsigned char, int): (*(volatile struct st_sci *)0xFFFFB0).SSR.BIT.TDRE	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(169,4):	SendSdOne	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SSR.BIT.TDRE = 0	×	ビットフィールドと範囲内の定数の比較なので問題ない。				
Drv¥DrvSdDmaSciItu.c(177,8):	SendSdOne	Operands of == have incompatible types (unsigned char, int): (*(volatile struct st_sci *)0xFFFFB0).SSR.BIT.RDRF	×	ビットフィールドと範囲内の定数の比較なので問題ない。				
Drv¥DrvSdDmaSciItu.c(181,5):	SendSdOne	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SSR.BIT.RDRF = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(323,3):	RecvSdKick	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RIE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(325,6):	RecvSdKick	Function startDmaShort expects arg 2 to be unsigned char gets int: ((0x08)) | (0) | (5)	×	サイズを意識した定数の#defineなので問題ない				
Drv¥DrvSdDmaSciItu.c(330,3):	RecvSdKick	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RIE = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(334,6):	RecvSdKick	Function startDmaShort expects arg 2 to be unsigned char gets int: ((0x10) | (0x08)) | (4)	×	サイズを意識した定数の#defineなので問題ない				
Drv¥DrvSdDmaSciItu.c(338,3):	RecvSdKick	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.TIE = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(368,3):	SendSdKick	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RIE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(370,6):	SendSdKick	Function startDmaShort expects arg 2 to be unsigned char gets int: ((0x10) | (0x08)) | (5)	×	サイズを意識した定数の#defineなので問題ない				
Drv¥DrvSdDmaSciItu.c(374,3):	SendSdKick	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RIE = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(378,6):	SendSdKick	Function startDmaShort expects arg 2 to be unsigned char gets int: ((0x08)) | (0) | (4)	×	サイズを意識した定数の#defineなので問題ない				
Drv¥DrvSdDmaSciItu.c(383,3):	SendSdKick	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.TIE = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(415,3):	RepeatSdKick	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RIE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(417,6):	RepeatSdKick	Function startDmaShort expects arg 2 to be unsigned char gets int: ((0x10) | (0x08)) | (5)	×	サイズを意識した定数の#defineなので問題ない				
Drv¥DrvSdDmaSciItu.c(421,3):	RepeatSdKick	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RIE = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(425,6):	RepeatSdKick	Function startDmaShort expects arg 2 to be unsigned char gets int: ((0x10) | (0x08)) | (4)	×	サイズを意識した定数の#defineなので問題ない				
Drv¥DrvSdDmaSciItu.c(429,3):	RepeatSdKick	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.TIE = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSciItu.c(455,4):	SdDmaPoll	Assignment of unsigned int to unsigned short int: *len = lastDmaLen - (*(volatile struct st_sam *)0xFFFF20).ETCR	×	ETCRがunsigned int となっている。サイズは2バイトであるということを確認しているため unsigned shortの方が望ましいが struct st_samはメーカー提供のレジスタ定義の構造体なので変更しない。				

ファイル名	関数名	解説	対応有無	理由	備考			
Drv¥DrvSdDmaSci¥tu.c(588,2):	startDmaShort	Implicitly only storage dmaAddr->MAR (type void *) not released before assignment: dmaAddr->MAR = pMem	×	動的なメモリ確保はおこなっていないので解放の問題は発生しない。また、startDmaShort()の呼び出し元で未使用状態であることを確認してから実行している。				
Drv¥DrvSdDmaSci¥tu.c(588,2):1	startDmaShort	Implicitly temp storage pMem assigned to implicitly only: dmaAddr->MAR = pMem	×	voidのポインタにcharのポインタを代入している。特に問題はない。				
Drv¥DrvSdDmaSci¥tu.c(592,2):	startDmaShort	Assignment of int to unsigned char: dmaAddr->DTCR.BIT.DTE = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(609,2):	CompleteDma0A	Assignment of int to unsigned char: (*(volatile struct st_sam *)0xFFFF20).DTCR.BIT.DTIE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(610,2):	CompleteDma0A	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RIE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(625,2):	CompleteDma0B	Assignment of int to unsigned char: (*(volatile struct st_sam *)0xFFFF28).DTCR.BIT.DTIE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(641,2):	InterruptRxi0	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SSR.BIT.RDRF = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(643,2):	InterruptRxi0	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.RIE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(658,2):	CompleteTxi0	Assignment of int to unsigned char: (*(volatile struct st_sci *)0xFFFFB0).SCR.BIT.TIE = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(690,2):	InitSdWait	Assignment of int to unsigned char: (*(volatile struct st_itu0 *)0xFFFF64).TCR.BIT.CCLR = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(691,2):	InitSdWait	Assignment of int to unsigned char: (*(volatile struct st_itu0 *)0xFFFF64).TCR.BIT.CKEG = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(692,2):	InitSdWait	Assignment of int to unsigned char: (*(volatile struct st_itu0 *)0xFFFF64).TCR.BIT.TPSC = 3	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(714,2):	SdWaitTimer	Assignment of int to unsigned char: (*(volatile struct st_itu *)0xFFFF60).TSTR.BIT.STR0 = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(717,11)	SdWaitTimer	Operands of == have incompatible types (unsigned char, int): (*(volatile struct st_itu0 *)0xFFFF64).TSR.BIT.IMFA	×	定数であり範囲内であることが明確なので問題ない				
Drv¥DrvSdDmaSci¥tu.c(718,4):	SdWaitTimer	Assignment of int to unsigned char: (*(volatile struct st_itu0 *)0xFFFF64).TSR.BIT.IMFA = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(721,3):	SdWaitTimer	Assignment of int to unsigned char: (*(volatile struct st_itu *)0xFFFF60).TSTR.BIT.STR0 = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(748,2):	InitSdPort	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFD4).DDR = 0x9f	×	サイズを意識した定数なので問題ない				
Drv¥DrvSdDmaSci¥tu.c(749,2):	InitSdPort	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFD4).DR.BYTE = 0xff	×	サイズを意識した定数なので問題ない				
Drv¥DrvSdDmaSci¥tu.c(750,2):	InitSdPort	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFD4).DR.BIT.B7 = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(769,6):	GetSdInsert	Operands of != have incompatible types (unsigned char, int): (*(volatile struct st_p1 *)0xFFFFD4).DR.BIT.B6 != 0	×	ビットフィールドとの比較で定数で範囲内であることは明確なので問題ない				
Drv¥DrvSdDmaSci¥tu.c(795,7):	GetSdProtect	Operands of == have incompatible types (unsigned char, int): (*(volatile struct st_p1 *)0xFFFFD4).DR.BIT.B5 == 0	×	ビットフィールドとの比較で定数で範囲内であることは明確なので問題ない				
Drv¥DrvSdDmaSci¥tu.c(820,3):	SetSdOnChipsel	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFD4).DR.BIT.B7 = 0	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
Drv¥DrvSdDmaSci¥tu.c(824,3):	SetSdOnChipsel	Assignment of int to unsigned char: (*(volatile struct st_p1 *)0xFFFFD4).DR.BIT.B7 = 1	×	ビットフィールドへの範囲内の定数設定なので問題は無い。				
..¥..¥FileSystem¥RfDriverInterface.h(27,2):		Struct tag struct t_rf_drv_ver defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	A function or variable is redefined. One of the declarations should use extern. (Use -redef to inhibit warning)	..¥..¥FileSystem¥¥RfDriverInterface.h(27,2): Previous definition of struct		
..¥..¥FileSystem¥RfDriverInterface.h(27,3):		Datatype T_RF_DRV_VER defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfDriverInterface.h(27,3): Previous definition of T_RF_DRV_VER			
..¥..¥FileSystem¥RfDriverInterface.h(57,2):		Struct tag struct s_rf_driver_interface defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfDriverInterface.h(57,2): Previous definition of struct			
..¥..¥FileSystem¥RfDriverInterface.h(57,3):		Datatype T_RF_DRIVER_INTERFACE defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfDriverInterface.h(57,3): Previous definition of			

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfInterface.h(27,2):		Struct tag struct t_rf_del_dir defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(27,2): Previous definition of struct t_rf_del_dir			
..¥..¥FileSystem¥RfInterface.h(27,3):		Datatype T_RF_DEL_DIR defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(27,3): Previous definition of T_RF_DEL_DIR			
..¥..¥FileSystem¥RfInterface.h(34,2):		Struct tag struct t_rf_del_file defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(34,2): Previous definition of struct t_rf_del_file			
..¥..¥FileSystem¥RfInterface.h(34,3):		Datatype T_RF_DEL_FILE defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(34,3): Previous definition of T_RF_DEL_FILE			
..¥..¥FileSystem¥RfInterface.h(45,2):		Struct tag struct t_rf_search defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(45,2): Previous definition of struct t_rf_search			
..¥..¥FileSystem¥RfInterface.h(45,3):		Datatype T_RF_SEARCH defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(45,3): Previous definition of T_RF_SEARCH			
..¥..¥FileSystem¥RfInterface.h(57,2):		Struct tag struct t_rf_search_dir defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(57,2): Previous definition of struct t_rf_search_dir			
..¥..¥FileSystem¥RfInterface.h(57,3):		Datatype T_RF_SEARCH_DIR defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(57,3): Previous definition of T_RF_SEARCH_DIR			
..¥..¥FileSystem¥RfInterface.h(71,2):		Struct tag struct t_rf_fcb defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(71,2): Previous definition of struct t_rf_fcb			
..¥..¥FileSystem¥RfInterface.h(71,2):1		Structure struct t_rf_fcb redeclared with fields { union !248 { ... } brk; unsigned char canWriteFlag; unsigned char canReadFlag;, ... }, previously declared with fields { union !4 { ... } brk; unsigned char canWriteFlag; unsigned char canReadFlag;, ... }	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。型の違う形式で再定義しているのは意識しておこなっている。	A struct, union or enum type is redefined with inconsistent fields or members. (Use -matchfields to inhibit warning)	..¥..¥FileSystem¥¥RfInterface.h(71,2): Previous declaration of struct t_rf_fcb	..¥..¥FileSystem¥RfInterface.h(66,4): Field brk redeclared as union !4 { ... }, previously declared as union !248 { ... }	..¥..¥FileSystem¥¥RfInterface.h(66,4): Previous declaration of brk
..¥..¥FileSystem¥RfInterface.h(71,3):		Datatype T_RF_FCB defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(71,3): Previous definition of T_RF_FCB			
..¥..¥FileSystem¥RfInterface.h(83,2):		Struct tag struct t_rf_time defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(83,2): Previous definition of struct t_rf_time			
..¥..¥FileSystem¥RfInterface.h(83,3):		Datatype T_RF_TIME defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(83,3): Previous definition of T_RF_TIME			
..¥..¥FileSystem¥RfInterface.h(93,2):		Struct tag struct t_fr_dir_info defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(93,2): Previous definition of struct t_fr_dir_info			
..¥..¥FileSystem¥RfInterface.h(93,3):		Datatype T_RF_DIR_INFO defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(93,3): Previous definition of T_RF_DIR_INFO			
..¥..¥FileSystem¥RfInterface.h(129,2):		Struct tag struct t_rf_drive_info defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(129,2): Previous definition of struct t_rf_drive_info			
..¥..¥FileSystem¥RfInterface.h(129,3):		Datatype T_RF_DRIVE_INFO defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(129,3): Previous definition of T_RF_DRIVE_INFO			
..¥..¥FileSystem¥RfInterface.h(140,2):		Struct tag struct t_rf_system_function defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(140,2): Previous definition of struct t_rf_system_function			
..¥..¥FileSystem¥RfInterface.h(140,3):		Datatype T_RF_SYSTEM_FUNCTION defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(140,3): Previous definition of T_RF_SYSTEM_FUNCTION			

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfInterface.h(199,2):		Struct tag struct t_rf_sys_ver defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(199,2): Previous definition of struct t_rf_sys_ver			
..¥..¥FileSystem¥RfInterface.h(199,3):		Datatype T_RF_FSYS_VER defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(199,3): Previous definition of T_RF_FSYS_VER			
..¥..¥FileSystem¥RfInterface.h(216,2):		Struct tag struct t_rf_file_buff defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(216,2): Previous definition of struct t_rf_file_buff			
..¥..¥FileSystem¥RfInterface.h(216,3):		Datatype T_RF_FILE_BUFF defined more than once	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。	..¥..¥FileSystem¥¥RfInterface.h(216,3): Previous definition of T_RF_FILE_BUFF			
..¥..¥FileSystem¥RfControlBuff.c(48,2):		Initial value of buffCtrlUnify.canWriteFlag is type int, expects unsigned char: (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定であればOKとする				
..¥..¥FileSystem¥RfControlBuff.c(49,2):		Initial value of buffCtrlUnify.fill is type int, expects unsigned char: 0	×	変数の初期値。Char のビットフィールドに0を設定している。範囲内の定数であることを確認した。またこの領域は未使用領域で				
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Assignment of int to unsigned char: buffCtrlUnify.canWriteFlag = (1)	×	ビットフィールドへ「TRUE」を設定している。定数で範囲内であることを確認した。				
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Assignment of int to unsigned char: buffCtrlUnify.canWriteFlag = (0)	×	ビットフィールドへ「FALSE」を設定している。定数で範囲内であることを確認した。				
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Assignment of int to unsigned char: buffCtrlUnify.canWriteFlag = (0)	×	ビットフィールドへ「FALSE」を設定している。定数で範囲内であることを確認した。				
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Implicitly only storage pBuffCtrl->buffAddr (type char *) not released before assignment: pBuffCtrl->buffAddr = &RfGenBuff[i * buffUnitSize]	×	buffAddrはこの初期化関数で初めて変更される。以前の使用は無い。				
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Immediate address &RfGenBuff[] assigned to implicitly only: pBuffCtrl->buffAddr = &RfGenBuff[i * buffUnitSize]	×	ユーザー定義のバッファを分割して使用するため既定のアドレスだが動的に設定する必要があるのでこの方法でよい。	An immediate address (result of & operator) is transferred inconsistently. (Use -immediatetrans to inhibit warning)			
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Assignment of int to unsigned char: pBuffCtrl->useFlag = (0)	×	ビットフィールドのサイズ内に収まる定数であることを確認した。フラグとして使用しているのでFALSEの設定は許可する。				
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Assignment of int to unsigned char: pBuffCtrl->updateFlag = (0)	×	ビットフィールドのサイズ内に収まる定数であることを確認した。フラグとして使用しているのでFALSEの設定は許可する。				
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Assignment of int to unsigned char: pBuffCtrl->justReadFlag = (0)	×	ビットフィールドのサイズ内に収まる定数であることを確認した。フラグとして使用しているのでFALSEの設定は許可する。				
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Assignment of int to unsigned char: pBuffCtrl->justWriteFlag = (0)	×	ビットフィールドのサイズ内に収まる定数であることを確認した。フラグとして使用しているのでFALSEの設定は許可する。				
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Function returns with possibly null storage derivable from global rfDriverInterface.write	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。	A possibly null pointer is reachable from a parameter or global variable that is not declared using a /*@null@*/ annotation. (Use -nullstate to inhibit warning)			
..¥..¥FileSystem¥RfControlBuff.	RfbInitBuff	Function returns with non-null global useDrvResource referencing null storage	×	使用中のリソースを示すポインタ。初期化でNULLにするのが意図通りなので問題ない。	A global variable does not satisfy its annotations when control is transferred. (Use -globstate to inhibit warning)	..¥..¥FileSystem¥RfControlBuff.c(174,19): Storage useDrvResource becomes null		
..¥..¥FileSystem¥RfControlBuff.	checkDriverFunct	Function returns with possibly null storage derivable from global rfDriverInterface.initialize	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。				
..¥..¥FileSystem¥RfControlBuff.	checkDriverFunct	Function returns with possibly null storage derivable from global rfDriverInterface.read	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。				
..¥..¥FileSystem¥RfControlBuff.	checkDriverFunct	Function returns with possibly null storage derivable from global rfDriverInterface.getBlockSize	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。				

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfControlBuff.	checkDriverFunct	Function returns with possibly null storage derivable from global rfDriverInterface.getCapacity	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。				
..¥..¥FileSystem¥RfControlBuff.	RfbPostSecureRe	Passed storage &insideBuffCtrl not completely defined: CheckInAddr (..., &insideBuffCtrl, ...)	×	初期化されていない。CheckInAddr()が初期化するのでここでの初期化は不要。	Storage derivable from a parameter, return value or global is not defined. Use /*@out@*/ to denote passed or returned storage which need not be defined. (Use -compdef to inhibit warning)			
..¥..¥FileSystem¥RfControlBuff.	RfbPostSecureRe	Function returns with possibly null storage derivable from global rfDriverInterface.readPoll	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。				
..¥..¥FileSystem¥RfControlBuff.	RfbPostSecureWr	Passed storage &insideBuffCtrl not completely defined: CheckInAddr (..., &insideBuffCtrl, ...)	×	初期化されていない。CheckInAddr()が初期化するのでここでの初期化は不要。				
..¥..¥FileSystem¥RfControlBuff.	RfbPostSecureWr	Function returns with possibly null storage derivable from global rfDriverInterface.write	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。				
..¥..¥FileSystem¥RfControlBuff.	RfbPostSecureWr	Function returns with possibly null storage derivable from global rfDriverInterface.writePoll	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。				
..¥..¥FileSystem¥RfControlBuff.	OneBuffTake	Passed storage &buffCtrl not completely defined: CheckInAddr (..., &buffCtrl, ...)	×	初期化されていない。CheckInAddr()が初期化するのでここでの初期化は不要。				
..¥..¥FileSystem¥RfControlBuff.	waitReadTarget	Operands of != have incompatible types (unsigned char, int): buffCtrl->justReadFlag != 0	×	範囲内の定数との比較であり問題ないことを確認した。				
..¥..¥FileSystem¥RfControlBuff.	waitReadTarget	Assignment of int to unsigned char: buffCtrl->justReadFlag = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfControlBuff.	waitReadTarget	Function returns with non-null global useDrvResource referencing null storage	×	useDrvResourceは使用中のバッファリソースのポインタが格納されるとともに未使用時はNULLを格納し未使用であることの判定にも使用しているのでNULLが格納されることは問題ない。	..¥..¥FileSystem¥RfControlBuff.c(426,22): Storage useDrvResource becomes null			
..¥..¥FileSystem¥RfControlBuff.	waitWriteTarget	Operands of != have incompatible types (unsigned char, int): buffCtrl->justWriteFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfControlBuff.	waitWriteTarget	Assignment of int to unsigned char: buffCtrl->justWriteFlag = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定であればOKとする				
..¥..¥FileSystem¥RfControlBuff.	waitWriteTarget	Function returns with non-null global useDrvResource referencing null storage	×	useDrvResourceは使用中のバッファリソースのポインタが格納されるとともに未使用時はNULLを格納し未使用であることの判定にも使用しているのでNULLが格納されることは問題ない。	..¥..¥FileSystem¥RfControlBuff.c(471,22): Storage useDrvResource becomes null			
..¥..¥FileSystem¥RfControlBuff.	waitResource	Function returns with non-null global useDrvResource referencing null storage	×	useDrvResourceは使用中のバッファリソースのポインタが格納されるとともに未使用時はNULLを格納し未使用であることの判定にも使用しているのでNULLが格納されることは問題ない。	..¥..¥FileSystem¥RfControlBuff.c(471,22): Storage useDrvResource may become null			
..¥..¥FileSystem¥RfControlBuff.	RfbWrite	Operands of != have incompatible types (unsigned char, int): buffCtrlUnify.canWriteFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定であればOKとする				
..¥..¥FileSystem¥RfControlBuff.	RfbFill	Operands of != have incompatible types (unsigned char, int): buffCtrlUnify.canWriteFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定であればOKとする				
..¥..¥FileSystem¥RfControlBuff.	OneBuffUpdate	Passed storage &buffCtrl not completely defined: CheckInAddr (..., &buffCtrl, ...)	×	初期化されていない。CheckInAddr()が初期化するのでここでの初期化は不要。				
..¥..¥FileSystem¥RfControlBuff.	OneBuffUpdate	Assignment of int to unsigned char: buffCtrl->updateFlag = (1)	×	1ビットのビットフィールドに1を設定している。範囲内の定数であり問題ないことを確				
..¥..¥FileSystem¥RfControlBuff.	OneBuffFill	Passed storage &buffCtrl not completely defined: CheckInAddr (..., &buffCtrl, ...)	×	初期化されていない。CheckInAddr()が初期化するのでここでの初期化は不要。				
..¥..¥FileSystem¥RfControlBuff.	OneBuffFill	Assignment of int to unsigned char: buffCtrl->updateFlag = (1)	×	1ビットのビットフィールドに1を設定している。範囲内の定数であり問題ないことを確				
..¥..¥FileSystem¥RfControlBuff.	FindInBuff	Operands of != have incompatible types (unsigned char, int): pBuffCtrl->useFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfControlBuff.	FindInBuff	Null storage retCode returned as non-null: retCode	×	NULLを戻り値として返すのは仕様であり問題ない。	Function returns a possibly null pointer, but is not declared using /*@null@*/ annotation of result. If function may return NULL, add /*@null@*/ annotation to the return value declaration. (Use -	..¥..¥FileSystem¥RfControlBuff.c(798,29): Storage retCode becomes null		
..¥..¥FileSystem¥RfControlBuff.	FindInBuff	Released storage buffCtrlUnify.buffCtrl reachable from global	×	テストで参照しているため global としている。その他ではglobalな参照はおこなわないこととしている。	..¥..¥FileSystem¥RfControlBuff.c(816,9): Storage buffCtrlUnify.buffCtrl released			
..¥..¥FileSystem¥RfControlBuff.	FindFreeBuff	Operands of == have incompatible types (unsigned char, int): pBuffCtrl->useFlag == (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfControlBuff.	FindFreeBuff	Null storage retCode returned as non-null: retCode	×	NULLを戻り値として返すのは仕様であり問題ない。	..¥..¥FileSystem¥RfControlBuff.c(832,29): Storage retCode becomes null			
..¥..¥FileSystem¥RfControlBuff.	FindFreeBuff	Released storage buffCtrlUnify.buffCtrl reachable from global	×	テストで参照しているため global としている。その他ではglobalな参照はおこなわないこととしている。	..¥..¥FileSystem¥RfControlBuff.c(848,9): Storage buffCtrlUnify.buffCtrl released			
..¥..¥FileSystem¥RfControlBuff.	FlushNear	Operands of != have incompatible types (unsigned char, int): pBuffCtrl->useFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfControlBuff.	FlushNear	Operands of != have incompatible types (unsigned char, int): pBuffCtrl->useFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfControlBuff.	FlushNear	Operands of != have incompatible types (unsigned char, int): pBuffCtrl->useFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfControlBuff.	FlushNear	Operands of != have incompatible types (unsigned char, int): targetCtrl->updateFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfControlBuff.	FlushNear	Only storage buffCtrlUnify.buffCtrl assigned to unqualified (targetCtrl aliases buffCtrlUnify.buffCtrl): *freeBuff = targetCtrl	×	動的メモリアクセスは使用しておらずメモリリークの心配は無い。当関数はstaticなので外部でこの領域を使用することもない。	The only reference to this storage is transferred to another reference (e.g., by returning it) that does not have the only annotation. This may lead to a memory leak, since the new reference is not necessarily released. (Use -onlytrans to			
..¥..¥FileSystem¥RfControlBuff.	reflectAll	Operands of != have incompatible types (unsigned char, int): pBuffCtrl->useFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfControlBuff.	reflectAll	Operands of != have incompatible types (unsigned char, int): pBuffCtrl->updateFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfControlBuff.	reflectABuffer	Operands of != have incompatible types (unsigned char, int): buffCtrl->updateFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfControlBuff.	reflectABuffer	Assignment of int to unsigned char: buffCtrl->updateFlag = (0)	×	buffCtrl->updateFlagは2値のフラグとして使用しており格納内容がFLASEの定数定義で範囲内なので問題ない。				
..¥..¥FileSystem¥RfControlBuff.	reflectABuffer	Assignment of int to unsigned char: buffCtrl->justWriteFlag = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfControlBuff.	ClearBuff	Assignment of int to unsigned char: buffCtrl->useFlag = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfControlBuff.	ClearBuff	Assignment of int to unsigned char: buffCtrl->updateFlag = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfControlBuff.	CheckInAddr	Arrow access from possibly null pointer insideBuffCtrl: insideBuffCtrl->buffSize	×	FlushNear() で insideBuffCtrl に値が格納されたことはFlushNear() の戻り値で確認しているため問題は無い。	A possibly null pointer is dereferenced. Value is either the result of a function which may return null (in which case, code should check it is not null), or a global, parameter or structure field declared with the null qualifier. (Use -nullderef to inhibit	..¥..¥FileSystem¥RfControlBuff.c(1104,20): Storage insideBuffCtrl may become null		
..¥..¥FileSystem¥RfControlBuff.	CheckInAddr	Assignment of int to unsigned char: insideBuffCtrl->useFlag = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfControlBuff.	CheckInAddr	Assignment of int to unsigned char: insideBuffCtrl->updateFlag = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfControlBuff.	CheckInAddr	Assignment of int to unsigned char: insideBuffCtrl->justReadFlag = 1	×	ビットフィールドのサイズ内に収まる定数であることを確認したので問題ない。 TRUE/FALSE の設定のみ許容する。				

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfControlBuff.	CheckInAddr	Fresh storage insideBuffCtrl not released before return	×	動的メモリ確保は使用していないためメモリリークは発生しない。	A memory leak has been detected. Storage allocated locally is not released before the last reference to it is lost. (Use -mustfreefresh to inhibit warning)	..¥..¥FileSystem¥RfControlBuff.c(1104,3): Fresh storage insideBuffCtrl		
..¥..¥FileSystem¥RfControlBuff.	physicalCopy	Operands of != have incompatible types (unsigned char, int): buffCtrlUnify.canWriteFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfControlBuff.	physicalCopy	Passed storage &pBuffCtrl not completely defined: CheckInAddr (..., &pBuffCtrl, ...)	×	CheckInAddr()で値が格納されるのでこれまでに初期化されていないことは問題な				
..¥..¥FileSystem¥RfControlBuff.	physicalCopy	Assignment of int to unsigned char: pBuffCtrl->updateFlag = (1)	×	ビットフィールドのサイズ内に収まる定数であることを確認したので問題ない。格納する値はTRUE/FLASEの宣言を使用する。				
..¥..¥FileSystem¥RfInterface.h(71,2):		Structure struct t_rf_fcb redeclared with fields { union !249 { ... } brk; unsigned char canWriteFlag; unsigned char canReadFlag;, ... }, previously declared with fields { union !248 { ... } brk; unsigned char canWriteFlag; unsigned char canReadFlag;, ... }	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。型の違う形式で再定義しているのは意識しておこなっている。	..¥..¥FileSystem¥¥RfInterface.h(71,2): Previous declaration of struct t_rf_fcb	..¥..¥FileSystem¥RfInterface.h(66,4): Field brk redeclared as union !248 { ... }, previously declared as union !249 { ... }	..¥..¥FileSystem¥RfInterface.h(66,4): Previous declaration of brk	
..¥..¥FileSystem¥RfControlBuff.h(18,16):		Function RfbInitBuff defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfControlBuff.c(178,1): Previous definition of RfbInitBuff			
..¥..¥FileSystem¥RfControlBuff.h(21,16):		Function RfbWrite defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfControlBuff.c(535,1): Previous definition of RfbWrite			
..¥..¥FileSystem¥RfControlBuff.h(27,16):		Function RfbFill defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfControlBuff.c(561,1): Previous definition of RfbFill			
..¥..¥FileSystem¥RfControlBuff.h(32,16):		Function RfbRead defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfControlBuff.c(226,1): Previous definition of RfbRead			
..¥..¥FileSystem¥RfControlBuff.h(37,16):		Function RfbPostSecureRead defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfControlBuff.c(250,1): Previous definition of RfbPostSecureRead			
..¥..¥FileSystem¥RfControlBuff.h(38,16):		Function RfbPostSecureWrite defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfControlBuff.c(279,1): Previous definition of RfbPostSecureWrite			
..¥..¥FileSystem¥RfControlBuff.h(44,16):		Function RfbReflect defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfControlBuff.c(954,1): Previous definition of RfbReflect			
..¥..¥FileSystem¥RfControlBuff.h(47,16):		Function RfbCopy defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfControlBuff.c(1162,1): Previous definition of RfbCopy			
..¥..¥FileSystem¥RfControlBuff.h(52,16):		Function RfbIsCanWrite defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfControlBuff.c(1233,1): Previous definition of RfbIsCanWrite			
..¥..¥FileSystem¥RfFormat.c(13	RfFormat	Passed storage formatInfo.tmp contains 11 undefined fields: driveSector, partitionSector, bpbAddr, fat1Point, ...	×	ここで実行しているcheckParm()で初期化するのでここまでで初期化されていないことは問題ない。				
..¥..¥FileSystem¥RfFormat.c(13	RfFormat	Passed storage formatInfo contains 7 undefined fields: bpbSector, resSector, bytesPerSector, sectPerClust, ...	×	ここで実行しているcheckParm()で初期化するのでここまでで初期化されていないことは問題ない。				
..¥..¥FileSystem¥RfFormat.c(16	checkParm	Passed storage driveInfo contains 16 undefined fields: fSystem, sectorPerCluster, bytesPerSector, bytesPerCluster, ...	×	RfmGetDriveInfo()は引数として渡している構造体の内容を取得する関数なのでここまでで初期されていないことは問題ない。				
..¥..¥FileSystem¥RfFormat.c(16	checkParm	Operands of == have incompatible types (unsigned char, int): driveInfo.drvInit == 0	×	範囲内の定数との比較であり問題ないことを確認した。				
..¥..¥FileSystem¥RfFormat.c(20	checkParm	Assignment of int to unsigned char: pFormatInfo->clearClust = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfFormat.c(20	checkParm	Assignment of int to unsigned char: pFormatInfo->clearClust = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfFormat.c(23	checkParmDos	Operands of == have incompatible types (unsigned char, int): pDrvInfo->mbrInit == (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfFormat.c(24	checkParmDos	Assignment of int to unsigned char: pFormatInfo->makeMbr = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfFormat.c(25	checkParmDos	Assignment of int to unsigned char: pFormatInfo->makeMbr = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfFormat.c(25	checkParmDos	Assignment of int to unsigned char: pFormatInfo->makeMbr = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfFormat.c(26	checkParmDos	Assignment of int to unsigned char: pFormatInfo->makeBpb = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfFormat.c(28	checkParmSd	Assignment of int to unsigned char: pFormatInfo->makeMbr = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfFormat.c(29	checkParmSd	Assignment of int to unsigned char: pFormatInfo->makeBpb = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfFormat.c(29	checkParmSd	Assignment of int to unsigned char: pFormatInfo->sectPerClust = 32	×	範囲内の定数であることを確認した。				
..¥..¥FileSystem¥RfFormat.c(29	checkParmSd	Assignment of int to unsigned char: pFormatInfo->sectPerClust = 64	×	範囲内の定数であることを確認した。				
..¥..¥FileSystem¥RfFormat.c(28	checkParmSd	Parameter pDrvInfo not used	×	checkParmMainte()、checkParmDos()を含めてシリーズ関数なので、checkParmSd()では使用していないが引数としては残しておく。	A function parameter is not used in the body of the function. If the argument is needed for type compatibility or future plans, use /*@unused@*/ in the argument declaration. (Use -paramuse to inhibit warning)			
..¥..¥FileSystem¥RfFormat.c(33	checkParmMainte	Operands of != have incompatible types (unsigned char, int): pDrvInfo->bpbInit != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfFormat.c(34	checkParmMainte	Assignment of int to unsigned char: pFormatInfo->makeMbr = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfFormat.c(34	checkParmMainte	Assignment of int to unsigned char: pFormatInfo->makeBpb = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfFormat.c(40	getDosClusterSiz	Assignment of int to unsigned char: pFormatInfo->sectPerClust = 8	×	定数であり範囲内であることを確認したので問題ない。				
..¥..¥FileSystem¥RfFormat.c(48	execFormat	Operands of != have incompatible types (unsigned char, int): pFormatInfo->makeMbr != (0)	×	1ビットのビットフィールドをフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfFormat.c(49	execFormat	Operands of != have incompatible types (unsigned char, int): pFormatInfo->makeBpb != (0)	×	1ビットのビットフィールドをフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfFormat.c(49	execFormat	Operands of != have incompatible types (unsigned char, int): pFormatInfo->makeMbr != (0)	×	1ビットのビットフィールドをフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfFormat.c(50	execFormat	Operands of != have incompatible types (unsigned char, int): pFormatInfo->clearClust != (0)	×	1ビットのビットフィールドをフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfFormat.c(58	makeMbr	Passed storage wkStr not completely defined (*wkStr is undefined): brokenULongData (..., wkStr)	×	charの配列をポインタとして使用している。問題ないと考える。				
..¥..¥FileSystem¥RfFormat.c(67	makeBpb	Passed storage wkStr not completely defined (*wkStr is undefined): brokenUShortData (..., wkStr)	×	charの配列をポインタとして使用している。問題ないと考える。				
..¥..¥FileSystem¥RfFormat.c(84	makeFat	Passed storage wkStr not completely defined (*wkStr is undefined): brokenUShortData (..., wkStr)	×	charの配列をポインタとして使用している。問題ないと考える。				
..¥..¥FileSystem¥RfFormat.c(85	makeFat	Passed storage wkStr not completely defined (*wkStr is undefined): brokenUShortData (..., wkStr)	×	charの配列をポインタとして使用している。問題ないと考える。				
..¥..¥FileSystem¥RfGeneral.c(32	rfMemSet	Implicitly temp storage dest returned as implicitly only: dest	×	受け取ったvoid型のポインタをそのままreturnしているので特に問題は無い。				
..¥..¥FileSystem¥RfGeneral.c(57	rfMemCpy	Implicitly temp storage dest returned as implicitly only: dest	×	受け取ったvoid型のポインタをそのままreturnしているので特に問題は無い。				
..¥..¥FileSystem¥RfInterface.h(71,2):		Structure struct t_rf_fcb redeclared with fields { union !250 { ... } brk; unsigned char canWriteFlag; unsigned char canReadFlag;, ... }, previously declared with fields { union !249 { ... } brk; unsigned char canWriteFlag; unsigned char canReadFlag;, ... }	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。型の違う形式で再定義しているのは意識しておこなっている。	..¥..¥FileSystem¥¥RfInterface.h(71,2): Previous declaration of struct t_rf_fcb	..¥..¥FileSystem¥RfInterface.h(66,4): Field brk redeclared as union !249 { ... }, previously declared as union !250 { ... }	..¥..¥FileSystem¥RfInterface.h(66,4): Previous declaration of brk	
..¥..¥FileSystem¥RfInterface.c(8	RfConclude	Function returns with possibly null storage derivable from global rfDriverInterface.conclude	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。				
..¥..¥FileSystem¥RfInterface.c(1	RfOpen	Operands of != have incompatible types (unsigned char, int): (mode & ((unsigned char)2)) != 0	×	定数であり範囲内であることは明確なので問題ない。				

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfInterface.c(1	RfOpen	Operands of != have incompatible types (unsigned char, int): (mode & ((unsigned char)8)) != 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(1	RfOpen	Operands of != have incompatible types (unsigned char, int): (pFcb->brk.sr.attribute & ((unsigned char)0x01)) != 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(1	RfOpen	Operands of != have incompatible types (unsigned char, int): (mode & ((unsigned char)4)) != 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(2	RfOpen	Operands of != have incompatible types (unsigned char, int): (mode & ((unsigned char)1)) != 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(2	RfOpen	Assignment of int to unsigned char: pFcb->canWriteFlag = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfInterface.c(2	RfOpen	Assignment of int to unsigned char: pFcb->useFlag = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfInterface.c(2	RfOpen	Null storage pFcb returned as non-null: pFcb	×	NULLを戻り値として返すのは仕様であり問題ない。	..¥..¥FileSystem¥RfInterface.c(123,20): Storage pFcb becomes null			
..¥..¥FileSystem¥RfInterface.c(2	openExist	Passed storage &err not completely defined: RfmGetParentDir (..., &err)	×	値を取得するための変数ポインタを渡しているの初期化していないのは問題ではな				
..¥..¥FileSystem¥RfInterface.c(2	openExist	Operands of != have incompatible types (unsigned char, int): (pFcb->brk.sr.attribute & ((unsigned char)0x10)) != 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(2	openExist	Operands of != have incompatible types (unsigned char, int): (pFcb->openMode & ((unsigned char)16)) != 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(3	openForNew	Return value (type unsigned long int) ignored: RfmFindFileEntry...	×	この場合は戻り値のエントリアドレスは使用しない				
..¥..¥FileSystem¥RfInterface.c(3	RfMkDir	Return value (type unsigned long int) ignored: RfmFindFileEntry...	×	この場合は戻り値のエントリアドレスは使用しない				
..¥..¥FileSystem¥RfInterface.c(4	RfDelFile	Operands of != have incompatible types (unsigned char, int): (pFcb->brk.sr.attribute & ((unsigned char)0x10)) != 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(4	RfDelFile	Fresh storage pFcb not released before return	×	動的なメモリアロケーションは使用していないので問題ない。また、RfmReleaseFcb()で pFcb の領域の解放を	..¥..¥FileSystem¥RfInterface.c(391,2): Fresh storage pFcb created			
..¥..¥FileSystem¥RfInterface.c(4	RfSetReadOnly	Fresh storage pFcb not released before return	×	動的なメモリアロケーションは使用していないので問題ない。また、RfmReleaseFcb()で pFcb の領域の解放を	..¥..¥FileSystem¥RfInterface.c(448,2): Fresh storage pFcb created			
..¥..¥FileSystem¥RfInterface.c(5	RfDelDir	Operands of == have incompatible types (unsigned char, int): (pFcb->brk.sr.attribute & ((unsigned char)0x10)) == 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(5	RfDelDir	Fresh storage pFcb not released before return	×	動的なメモリアロケーションは使用していないので問題ない。また、RfmReleaseFcb()で pFcb の領域の解放を	..¥..¥FileSystem¥RfInterface.c(515,2): Fresh storage pFcb created			
..¥..¥FileSystem¥RfInterface.c(6	RfMove	Operands of != have incompatible types (unsigned char, int): (pFcb->brk.sr.attribute & ((unsigned char)0x10)) != 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(6	RfMove	Passed storage moveInfo contains 1 undefined field: fill	×	fillはアライメント調整のために挿入した未使用領域なので初期化されていないことは問題ない。				
..¥..¥FileSystem¥RfInterface.c(6	RfMove	Passed storage moveInfo contains 1 undefined field: fill	×	fillはアライメント調整のために挿入した未使用領域なので初期化されていないことは問題ない。				
..¥..¥FileSystem¥RfInterface.c(6	RfMove	Fresh storage pFcb not released before return	×	動的なメモリアロケーションは使用していないので問題ない。また、RfmReleaseFcb()で pFcb の領域の解放を	..¥..¥FileSystem¥RfInterface.c(579,2): Fresh storage pFcb created			
..¥..¥FileSystem¥RfInterface.c(7	RfFlush	Operands of != have incompatible types (unsigned char, int): pFcb->useFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfInterface.c(7	RfGetSize	Operands of != have incompatible types (unsigned char, int): pFcb->useFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfInterface.c(7	RfGetSize	Operands of == have incompatible types (unsigned char, int): (pFcb->brk.sr.attribute & ((unsigned char)0x10)) == 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(8	RfGetVerSion	Function returns with possibly null storage derivable from global rfDriverInterface.getVer	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。				
..¥..¥FileSystem¥RfInterface.c(8	RfOpenDir	Operands of == have incompatible types (unsigned char, int): (pFcb->brk.sd.attribute & ((unsigned char)0x10)) == 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfInterface.c(8	RfOpenDir	Assignment of int to unsigned char: pFcb->useFlag = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfInterface.c(8	RfOpenDir	Fresh storage pFcb (type T_RF_FCB *) not released before assignment: pFcb = NULL	×	動的なメモリアロケーションは使用していないので問題ない。	..¥..¥FileSystem¥RfInterface.c(848,2): Fresh storage pFcb created			
..¥..¥FileSystem¥RfInterface.c(8	RfOpenDir	Possibly null storage pFcb returned as non-null: pFcb	×	エラー発生時 NULL を返す仕様なのでこのままで良い。	..¥..¥FileSystem¥RfInterface.c(877,10): Storage pFcb may become null			
..¥..¥FileSystem¥RfInterface.h(71,2):		Structure struct t_rf_fcb redeclared with fields { union !251 { ... } brk; unsigned char canWriteFlag; unsigned char canReadFlag;, ... }, previously declared with fields { union !250 { ... } brk; unsigned char canWriteFlag; unsigned char canReadFlag;, ... }	×	ヘッダファイルに二重定義防止の#ifndefが入っており、指摘されている定義先も同一箇所なので問題ない。型の違う形式で再定義しているのは意識しておこなっている。	..¥..¥FileSystem¥RfInterface.h(71,2): Previous declaration of struct t_rf_fcb	..¥..¥FileSystem¥RfInterface.h(66,4): Field brk redeclared as union !250 { ... }, previously declared as union !251 { ... }	..¥..¥FileSystem¥RfInterface.h(66,4): Previous declaration of brk	
..¥..¥FileSystem¥RfInterface.h(228,16):		Function RfInitFileSystem defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(51,1): Previous definition of RfInitFileSystem			
..¥..¥FileSystem¥RfInterface.h(230,16):		Function RfConclude defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(84,1): Previous definition of RfConclude			
..¥..¥FileSystem¥RfInterface.h(232,16):		Function RfGetDriveInfo defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(106,1): Previous definition of RfGetDriveInfo			
..¥..¥FileSystem¥RfInterface.h(238,12):		Function RfOpen defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(245,1): Previous definition of RfOpen			
..¥..¥FileSystem¥RfInterface.h(240,16):		Function RfWrite defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(720,1): Previous definition of RfWrite			
..¥..¥FileSystem¥RfInterface.h(242,16):		Function RfRead defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(676,1): Previous definition of RfRead			
..¥..¥FileSystem¥RfInterface.h(244,16):		Function RfSeek defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(740,1): Previous definition of RfSeek			
..¥..¥FileSystem¥RfInterface.h(246,16):		Function RfFlush defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(767,1): Previous definition of RfFlush			
..¥..¥FileSystem¥RfInterface.h(248,16):		Function RfGetSize defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(802,1): Previous definition of RfGetSize			
..¥..¥FileSystem¥RfInterface.h(250,16):		Function RfClose defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(658,1): Previous definition of RfClose			
..¥..¥FileSystem¥RfInterface.h(256,16):		Function RfMkDir defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(365,1): Previous definition of RfMkDir			
..¥..¥FileSystem¥RfInterface.h(258,16):		Function RfDelFile defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(424,1): Previous definition of RfDelFile			
..¥..¥FileSystem¥RfInterface.h(260,16):		Function RfDelDir defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(551,1): Previous definition of RfDelDir			
..¥..¥FileSystem¥RfInterface.h(262,16):		Function RfMove defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(631,1): Previous definition of RfMove			
..¥..¥FileSystem¥RfInterface.h(264,16):		Function RfSetReadOnly defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(486,1): Previous definition of RfSetReadOnly			
..¥..¥FileSystem¥RfInterface.h(266,16):		Function RfGetVerSion defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(826,1): Previous definition of RfGetVerSion			
..¥..¥FileSystem¥RfInterface.h(268,12):		Function RfOpenDir defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(880,1): Previous definition of RfOpenDir			
..¥..¥FileSystem¥RfInterface.h(270,16):		Function RfReadDir defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(904,1): Previous definition of RfReadDir			
..¥..¥FileSystem¥RfInterface.h(272,16):		Function RfCloseDir defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfInterface.c(920,1): Previous definition of RfCloseDir			
..¥..¥FileSystem¥RfGeneral.h(18,7):		Function rfMemSet defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfGeneral.c(33,1): Previous definition of rfMemSet			
..¥..¥FileSystem¥RfGeneral.h(20,7):		Function rfMemCpy defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfGeneral.c(58,1): Previous definition of rfMemCpy			
..¥..¥FileSystem¥RfGeneral.h(22,5):		Function rfMemCmp defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfGeneral.c(93,1): Previous definition of rfMemCmp			
..¥..¥FileSystem¥RfGeneral.h(24,5):		Function rfStrLen defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfGeneral.c(115,1): Previous definition of rfStrLen			
..¥..¥FileSystem¥RfGeneral.h(26,5):		Function rfToUpper defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfGeneral.c(136,1): Previous definition of rfToUpper			
..¥..¥FileSystem¥RfGeneral.h(28,5):		Function rfStrCpy defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfGeneral.c(164,1): Previous definition of rfStrCpy			
..¥..¥FileSystem¥RfGeneral.h(30,5):		Function rfStrCat defined more than once	×	ヘッダで型宣言をしている	..¥..¥FileSystem¥RfGeneral.c(196,1): Previous definition of rfStrCat			

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfMiddle.c(210	clearSystemVariai	Assignment of int to unsigned char: rfDriveInfo.drvInit = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfMiddle.c(211	clearSystemVariai	Assignment of int to unsigned char: rfDriveInfo.mbrInit = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfMiddle.c(212	clearSystemVariai	Assignment of int to unsigned char: rfDriveInfo.bpbInit = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfMiddle.c(235	RfmIsInitConfirm	Operands of != have incompatible types (unsigned char, int): rfDriveInfo.bpbInit != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfMiddle.c(285	makeDriveInfo	Assignment of int to unsigned char: rfDriveInfo.drvInit = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfMiddle.c(298	makeDriveInfo	Passed storage buff not completely defined (*buff is undefined): RfbRead (..., buff, ...)	×	RfbRead() で読み取った値を格納する領域なので初期化の必要は無い				
..¥..¥FileSystem¥RfMiddle.c(389	makeDriveInfo	Assignment of int to unsigned char: rfDriveInfo.bpbInit = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfMiddle.c(429	judgeMbr	Variable systemId initialized to type int, expects char: 0	×	範囲内の定数であることを確認した。				
..¥..¥FileSystem¥RfMiddle.c(432	judgeMbr	Passed storage buff not completely defined (*buff is undefined): RfbRead (..., buff, ...)	×	RfbRead() で読み取った値を格納する領域なので初期化の必要は無い				
..¥..¥FileSystem¥RfMiddle.c(455	judgeMbr	Assignment of int to unsigned char: rfDriveInfo.mbrInit = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfMiddle.c(457	judgeMbr	Operands of != have incompatible types (char, int): systemId != 0	×	範囲内の定数なので許容する。				
..¥..¥FileSystem¥RfMiddle.c(461	judgeMbr	Assignment of int to unsigned char: rfDriveInfo.mbrInit = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfMiddle.c(495	RfmGetDriveInfo	Return value (type void *) ignored: rfMemCpy(driveIn...	×	戻り値は使用しないので見ない。rfMemCpy()の型は標準関数のmemcpy()に合わせているので戻り値を使用しないがvoidとはしない。	Result returned by function call is not used. If this is intended, can cast result to (void) to eliminate message. (Use -retvalother to inhibit warning)			
..¥..¥FileSystem¥RfMiddle.c(518	RfmFindFreeFcb	Operands of == have incompatible types (unsigned char, int): rfFileControlBlock[i].useFlag == (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfMiddle.c(524	RfmFindFreeFcb	Immediate address &rfFileControlBlock[] returned as implicitly only (pFcb aliases &rfFileControlBlock[]): pFcb	×	rfFileControlBlockのポインタを返す仕様なので問題ない。				
..¥..¥FileSystem¥RfMiddle.c(524	RfmFindFreeFcb	Null storage pFcb returned as non-null: pFcb	×	pFcb が NULL のまま return する場合はエラーを意味するので問題ない。	..¥..¥FileSystem¥RfMiddle.c(513,20): Storage pFcb becomes null			
..¥..¥FileSystem¥RfMiddle.c(545	RfmCheckAlready	Operands of != have incompatible types (unsigned char, int): rfFileControlBlock[i].useFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfMiddle.c(573	RfmFindUseFcb	Operands of != have incompatible types (unsigned char, int): rfFileControlBlock[i].useFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfMiddle.c(579	RfmFindUseFcb	Immediate address &rfFileControlBlock[] returned as implicitly only (pFcb aliases &rfFileControlBlock[]): pFcb	×	rfFileControlBlockのポインタを返す仕様で問題ない。				
..¥..¥FileSystem¥RfMiddle.c(579	RfmFindUseFcb	Null storage pFcb returned as non-null: pFcb	×	pFcb が NULL のまま return する場合はエラーを意味するので問題ない。	..¥..¥FileSystem¥RfMiddle.c(568,20): Storage pFcb becomes null			
..¥..¥FileSystem¥RfMiddle.c(598	InitAllFcb	Assignment of int to unsigned char: rfFileControlBlock[i].useFlag = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfMiddle.c(624	RfmInitUseFcb	Assignment of int to unsigned char: pFcb->useFlag = (1)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				
..¥..¥FileSystem¥RfMiddle.c(626	RfmInitUseFcb	Assignment of int to unsigned char: pFcb->openMode = 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfMiddle.c(656	RfmReleaseFcb	Operands of != have incompatible types (unsigned char, int): pFcb->useFlag != (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfMiddle.c(659	RfmReleaseFcb	Assignment of int to unsigned char: pFcb->useFlag = (0)	×	1ビットのビットフィールドでフラグとして使用しているのでTRUE/FALSEの設定のみを許容する。				

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfMiddle.c(683	RfmFlush	Operands of != have incompatible types (unsigned char, int): (pFcb->openMode & (((unsigned char)2) | ((unsigned char)4) | ((unsigned char)8))) != 0	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfMiddle.c(727	RfmTakeExistInfo	Passed storage buff not completely defined (*buff is undefined): RfbRead (..., buff, ...)	×	RfbRead() で読み取った値を格納する領域なので初期化の必要は無い				
..¥..¥FileSystem¥RfMiddle.c(776	RfmRead	Variable overSize initialized to type int, expects char: (0)	×	フラグとして使用しているので TRUE/FALSEの設定、判定は許容する				
..¥..¥FileSystem¥RfMiddle.c(788	RfmRead	Assignment of int to char: overSize = (1)	×	フラグとして使用しているので TRUE/FALSEの設定、判定は許容する				
..¥..¥FileSystem¥RfMiddle.c(832	RfmRead	Operands of != have incompatible types (char, int): overSize != (0)	×	フラグとして使用しているので TRUE/FALSEの設定、判定は許容する				
..¥..¥FileSystem¥RfMiddle.c(106	RfmWriteOver	Variable overSize initialized to type int, expects char: (0)	×	フラグとして使用しているので TRUE/FALSEの設定、判定は許容する				
..¥..¥FileSystem¥RfMiddle.c(106	RfmWriteOver	Assignment of int to char: overSize = (1)	×	フラグとして使用しているので TRUE/FALSEの設定、判定は許容する				
..¥..¥FileSystem¥RfMiddle.c(111	RfmWriteOver	Operands of != have incompatible types (char, int): overSize != (0)	×	フラグとして使用しているので TRUE/FALSEの設定、判定は許容する				
..¥..¥FileSystem¥RfMiddle.c(114	clusterToAddr	Operands of < have incompatible types (unsigned short int, int): cluster < (rfDriveInfo.clusterNum + 2)	×	右辺、左辺の変数はともに unsigned short であることを確認した。				
..¥..¥FileSystem¥RfMiddle.c(118	fillCluster	Function returns with possibly null storage derivable from global rfDriverInterface.fill	×	rfDriverInterface はドライバ関数を登録する構造体でメンバーが null であっても内容を変更してはいけない。				
..¥..¥FileSystem¥RfMiddle.c(123	RfmSeek	Operands of == have incompatible types (unsigned char, int): pFcb->canWriteFlag == (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfMiddle.c(127	RfmSeek	Operands of == have incompatible types (unsigned char, int): pFcb->canWriteFlag == (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfMiddle.c(130	RfmSeek	Operands of == have incompatible types (unsigned char, int): pFcb->canWriteFlag == (0)	×	1ビットのビットフィールドでフラグとして使用しているのでFALSEとの比較だけを許可				
..¥..¥FileSystem¥RfMiddle.c(141	RfmGetParentDir	Assignment of int to unsigned short int: pathEnd = bodyTop - 1	×	右辺、左辺の変数はともに unsigned short であることを確認した。また、bodyTopがゼロの場合はここを通らないようチェックして				
..¥..¥FileSystem¥RfMiddle.c(147	findLastDir	Passed storage takeBuff not completely defined (*takeBuff is undefined): RfbRead (..., takeBuff, ...)	×	RfbRead() で読み取った値を格納する領域なので初期化の必要は無い				
..¥..¥FileSystem¥RfMiddle.c(155	findShortName	Passed storage findName not completely defined (*findName is undefined): checkShortName (..., findName,	×	findName は checkShortName() の結果が格納される領域なので初期化の必要は無				
..¥..¥FileSystem¥RfMiddle.c(155	findShortName	Null storage passed as non-null param: checkShortName (..., NULL)	×	checkShortName()の第3引数はファイル名の書式を格納する領域をしているする仕様 NULL の際は使用しないこととなっていることを確認した。	A possibly null pointer is passed as a parameter corresponding to a formal parameter with no /*@null@*/ annotation. If NULL may be used for this parameter, add a /*@null@*/ annotation to the function parameter declaration. (Use -nullpass to inhibit warning)			
..¥..¥FileSystem¥RfMiddle.c(156	findShortName	Passed storage searchDirInfo contains 6 undefined fields: entryAddr, firstCluster, fill, fill2, ...	×	searchDirInfo のメンバーのうち curCluster,pointInCluster以外は未使用領域、または答えが格納される領域なので初期化の必要は無い。				
..¥..¥FileSystem¥RfMiddle.c(157	findShortName	Passed storage inDirName not completely defined (*inDirName is undefined): RfbRead (..., inDirName, ...)	×	RfbRead() で読み取った値を格納する領域なので初期化の必要は無い				
..¥..¥FileSystem¥RfMiddle.c(165	checkShortName	Return value (type unsigned short int) ignored: takeToken(0, flN...	×	takeToken()の戻り値はトークンの先頭位置、takeToken()の第三引数にはトークンの長さが設定される。先頭位置はここでは使用しないので問題ない。				
..¥..¥FileSystem¥RfMiddle.c(165	checkShortName	Return value (type void *) ignored: rfMemCpy(findNam...	×	rfMemCpy()の戻り値は複写先のポインタだがここでは使用しないので見なくてもよ				
..¥..¥FileSystem¥RfMiddle.c(166	checkShortName	Return value (type void *) ignored: rfMemSet(&findNa...	×	rfMemSet()の戻り値は複写先のポインタだがここでは使用しないので見なくてもよい。				
..¥..¥FileSystem¥RfMiddle.c(167	checkShortName	Return value (type unsigned short int) ignored: takeToken(0, &fl...	×	ここではトークンの長さだけが必要で戻り値であるトークンの先頭位置は使用しないので参照していない。				
..¥..¥FileSystem¥RfMiddle.c(167	checkShortName	Return value (type void *) ignored: rfMemCpy(&findNa...	×	rfMemCpy()の戻り値は複写先のポインタだがここでは使用しないので見なくてもよ				
..¥..¥FileSystem¥RfMiddle.c(168	checkShortName	Return value (type void *) ignored: rfMemSet(&findNa...	×	rfMemSet()の戻り値は複写先のポインタだがここでは使用しないので見なくてもよい。				

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfMiddle.c(169	checkShortName	Assignment of int to char: *charType = 0	×	範囲内の定数であることが明確なので問題ない。				
..¥..¥FileSystem¥RfMiddle.c(174	checkException	Return value (type void *) ignored: rfMemSet(findNam...	×	rfMemSet()の戻り値は複写先のポインタだがここでは使用しないので見なくてもよい。				
..¥..¥FileSystem¥RfMiddle.c(178	RfmCreateFileEnt	Passed storage &err not completely defined: findFreeEntry (..., &err)	×	err は findFreeEntry() が設定するので初期化不要				
..¥..¥FileSystem¥RfMiddle.c(178	RfmCreateFileEnt	Function makeEntry expects arg 3 to be unsigned char gets int: 0	×	定数であり常に範囲内であることを確認した				
..¥..¥FileSystem¥RfMiddle.c(181	RfmCreateDirEntr	Passed storage &err not completely defined: findFreeClust (&err)	×	err は findFreeClust() が設定するので初期化不要				
..¥..¥FileSystem¥RfMiddle.c(182	RfmCreateDirEntr	Function fillCluster expects arg 2 to be char gets int: 0x00	×	範囲内の定数であることが明確なので問題ない				
..¥..¥FileSystem¥RfMiddle.c(199	getNextEntryNarr	Passed storage &buffNameTop not completely defined: RfbRead (..., &buffNameTop, ...)	×	この RfbRead で buffName に値を設定するのでここまでで何も設定されていないことは問題ではない。				
..¥..¥FileSystem¥RfMiddle.c(207	findFreeEntry	Passed storage searchDirInfo contains 6 undefined fields: entryAddr, firstCluster, fill, fill2, ...	×	searchDirInfo のメンバーのうち curCluster,pointInCluster以外は未使用領域、または答えが格納される領域なので初期化の必要は無い。				
..¥..¥FileSystem¥RfMiddle.c(209	findFreeEntry	Function fillCluster expects arg 2 to be char gets int: 0x00	×	定数であり範囲内であることが明確なので問題ない。				
..¥..¥FileSystem¥RfMiddle.c(210	findFreeEntry	Field searchDirInfo.entryAddr used before definition	×	前のwhile()は必ず1回は通るようになっておりその中で確実に初期化される	An rvalue is used that may not be initialized to a value on some execution path. (Use -usedef to inhibit warning)			
..¥..¥FileSystem¥RfMiddle.c(217	RfmDelDir	Passed storage searchDirInfo contains 6 undefined fields: entryAddr, firstCluster, fill, fill2, ...	×	searchDirInfo のメンバーのうち curCluster,pointInCluster以外は未使用領域、または答えが格納される領域なので初期化の必要は無い。				
..¥..¥FileSystem¥RfMiddle.c(219	RfmDelDir	Assignment of int to unsigned short int: pFcb->brk.dd.nest = orgInfo.nest + 1	×	定数であり範囲内であることが明確である				
..¥..¥FileSystem¥RfMiddle.c(226	RfmCheckRoots	Passed storage strCluster not completely defined (*strCluster is undefined): RfbRead (..., strCluster, ...)	×	この RfbRead で strCluster に値を設定するのでここまでで何も設定されていないことは問題ではない。				
..¥..¥FileSystem¥RfMiddle.c(230	RfmMoveEntry	Passed storage dirWork not completely defined (*dirWork is undefined): RfbRead (..., dirWork, ...)	×	この RfbRead で dirWork に値を設定するのでここまでで何も設定されていないことは問題ではない。				
..¥..¥FileSystem¥RfMiddle.c(232	RfmMoveEntry	Passed storage saveCluster not completely defined (*saveCluster is undefined): brokenUShortData (..., saveCluster)	×	brokenUShortData()で saveCluster に設定するので初期化されていないことは問題ではない				
..¥..¥FileSystem¥RfMiddle.c(238	makeEntry	Variable nameChar initialized to type int, expects char: 0	×	定数であり範囲内であることが明確であることを確認したので問題ない				
..¥..¥FileSystem¥RfMiddle.c(239	makeEntry	Passed storage sepName not completely defined (*sepName is undefined): checkShortName (..., sepName, ...)	×	sepName は checkShortName() で設定されるので初期化されていないことは問題ではない				
..¥..¥FileSystem¥RfMiddle.c(241	makeEntry	Passed storage nowTime contains 7 undefined fields: year, month, day, hour, ...	×	時刻を取得する関数を実行して nowTime 構造体のメンバー変数を取得するので初期化されていないことは問題ではない				
..¥..¥FileSystem¥RfMiddle.c(242	makeEntry	Passed storage dirTime not completely defined (*dirTime is undefined): toDirTime (dirTime, ...)	×	toDirTime() で dirTime を設定するので初期化されていないことは問題ない				
..¥..¥FileSystem¥RfMiddle.c(245	makeEntry	Passed storage saveCluster not completely defined (*saveCluster is undefined): brokenUShortData (..., saveCluster)	×	brokenUShortData()で saveCluster に設定するので初期化されていないことは問題ではない				
..¥..¥FileSystem¥RfMiddle.c(246	makeEntry	Function returns with possibly null storage derivable from global rfSystemFunction.pGetNowTime	×	rfSystemFunction.pGetNowTime はシステムインプリメントの際に日付取得関数を定義する領域なのでNULLであってもこの関数で何かを設定する必要はない。				
..¥..¥FileSystem¥RfMiddle.c(249	updateEntry	Passed storage nowTime contains 7 undefined fields: year, month, day, hour, ...	×	時刻を取得する関数を実行して nowTime 構造体のメンバー変数を取得するので初期化されていないことは問題ではない				
..¥..¥FileSystem¥RfMiddle.c(249	updateEntry	Passed storage buff not completely defined (*buff is undefined): toDirTime (buff, ...)	×	toDirTime() で buff を設定するので初期化されていないことは問題ない				
..¥..¥FileSystem¥RfMiddle.c(251	updateEntry	Passed storage buff not completely defined (*buff is undefined): brokenUShortData (..., buff)	×	buff brokenUShortData() で設定するので初期化されていないことは問題ではない				

ファイル名	関数名	解説	対応有無	理由	備考			
..¥..¥FileSystem¥RfMiddle.c(252	updateEntry	Function returns with possibly null storage derivable from global rfSystemFunction.pGetNowTime	×	rfSystemFunction.pGetNowTime はシステムインプリメントの際に日付取得関数を定義する領域なのでNULLであってもこの関数で何かを設定する必要はない。				
..¥..¥FileSystem¥RfMiddle.c(260	toDirTime	Left operand of << may be negative (int): (pOrgTime->year - 1980) << 9	×	直前の判定で範囲チェックしてオーバーフローしないことは保障されている。左オペランドの wordWk は unsigned の変数で shift 動作により符号が反転することの影響はな	The left operand to a shift operator may be negative (behavior is implementation-defined). (Use -shiftimplementation to inhibit warning)			
..¥..¥FileSystem¥RfMiddle.c(263	fromDirCreateTim	Assignment of int to unsigned short int: pRfTime->mSec = ((unsigned short int)(*dirTime) % 100) * 10	×	範囲内になることが確実な定数との演算であり型違いの問題は発生しない				
..¥..¥FileSystem¥RfMiddle.c(265	fromDirUpdateTim	Assignment of int to unsigned short int: pRfTime->hour = (wordWk >> 11) & 0x1f	×	使用している int の定数は範囲内であることが明確なので問題ない				
..¥..¥FileSystem¥RfMiddle.c(265	fromDirUpdateTim	Assignment of int to unsigned short int: pRfTime->min = (wordWk >> 5) & 0x3f	×	使用している int の定数は範囲内であることが明確なので問題ない				
..¥..¥FileSystem¥RfMiddle.c(265	fromDirUpdateTim	Assignment of int to unsigned short int: pRfTime->sec = (wordWk & 0x1f) * 2	×	使用している int の定数は範囲内であることが明確なので問題ない				
..¥..¥FileSystem¥RfMiddle.c(268	fromDirAccessDat	Assignment of int to unsigned short int: pRfTime->year = (wordWk >> 9) + 1980	×	使用している int の定数は範囲内であることが明確なので問題ない				
..¥..¥FileSystem¥RfMiddle.c(268	fromDirAccessDat	Assignment of int to unsigned short int: pRfTime->month = (wordWk >> 5) & 0x0f	×	使用している int の定数は範囲内であることが明確なので問題ない				
..¥..¥FileSystem¥RfMiddle.c(268	fromDirAccessDat	Assignment of int to unsigned short int: pRfTime->day = wordWk & 0x1f	×	使用している int の定数は範囲内であることが明確なので問題ない				
..¥..¥FileSystem¥RfMiddle.c(276	findFreeClust	Assignment of int to unsigned short int: freeClust = (freeClust + 1) % (rfDriveInfo.clusterNum + 2)	×	計算に使用する2つの変数は unsigned short でオーバーフロー以外では負になる計算はおこなっていないので問題ない。クラスタ番号の計算なのでオーバーフローす				
..¥..¥FileSystem¥RfMiddle.c(282	getFatVal	Operands of < have incompatible types (unsigned short int, int): cluster < (rfDriveInfo.clusterNum + 2)	×	定数であり範囲内であることは明確なので問題ない。				
..¥..¥FileSystem¥RfMiddle.c(287	getFat12Value	Passed storage buff not completely defined (*buff is undefined): RfbRead (..., buff, ...)	×	この RfbRead で buff に値を設定するのでここまでで何も設定されていないことは問題ではない。				
..¥..¥FileSystem¥RfMiddle.c(291	getFat16Value	Passed storage buff not completely defined (*buff is undefined): RfbRead (..., buff, ...)	×	この RfbRead で buff に値を設定するのでここまでで何も設定されていないことは問題ではない。				
..¥..¥FileSystem¥RfMiddle.c(300	setFat12Value	Passed storage buff not completely defined (*buff is undefined): RfbRead (..., buff, ...)	×	この RfbRead で buff に値を設定するのでここまでで何も設定されていないことは問題ではない。				
..¥..¥FileSystem¥RfMiddle.c(307	setFat16Value	Passed storage buff not completely defined (*buff is undefined): brokenUShortData (..., buff)	×	buff は brokenUShortData() で設定するので初期化されていないことは問題ではない				
..¥..¥FileSystem¥RfMiddle.c(338	takeLastToken	Assignment of int to unsigned short int: startPos = i + 1	×	i と startPos の型は一致している。i に加える値は定数であり範囲内であることが明確なので問題はない。				
..¥..¥FileSystem¥RfMiddle.c(348	isValidShortChar	Operands of >= have incompatible types (char, int): (name[i] & 0xff) >= 0x80	×	型が異なるがサイズの問題が発生しないようマスクして比較している				
..¥..¥FileSystem¥RfMiddle.c(348	isValidShortChar	Operands of <= have incompatible types (char, int): (name[i] & 0xff) <= 0x8f	×	型が異なるがサイズの問題が発生しないようマスクして比較している				
..¥..¥FileSystem¥RfMiddle.c(348	isValidShortChar	Operands of >= have incompatible types (char, int): (name[i] & 0xff) >= 0xe0	×	型が異なるがサイズの問題が発生しないようマスクして比較している				
..¥..¥FileSystem¥RfMiddle.c(348	isValidShortChar	Operands of <= have incompatible types (char, int): (name[i] & 0xff) <= 0xff	×	型が異なるがサイズの問題が発生しないようマスクして比較している				
..¥..¥FileSystem¥RfMiddle.c(349	isValidShortChar	Operands of < have incompatible types (unsigned short int, arbitrary unsigned integral type): j <	×	どちらも符号なしで比較されているので問題ない				
..¥..¥FileSystem¥RfMiddle.c(353	RfmReadDir	Passed storage entry not completely defined (*entry is undefined): RfbRead (..., entry, ...)	×	この RfbRead で entry に値を設定するのでここまでで何も設定されていないことは問題ではない。				